特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2003年12月24日 (24.12.2003)

PCT

(10) 国際公開番号
WO 03/107182 A1

(51) 国際特許分類[7]: G06F 9/445

(21) 国際出願番号: PCT/JP03/07390

(22) 国際出願日: 2003年6月11日 (11.06.2003)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願2002-171338 2002年6月12日 (12.06.2002) JP

(71) 出願人 (米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) [JP/JP]; 〒571-8501 大阪府 門真市 大字門真１００６ Osaka (JP).

(72) 発明者; および
(75) 発明者/出願人 (米国についてのみ): 片岡 充照 (KATAOKA,Mitsuteru) [JP/JP]; 〒576-0034 大阪府 交野市 天野が原町４－１５－１１ Osaka (JP).

(74) 代理人: 小笠原 史朗 (OGASAWARA,Shiro); 〒564-0053 大阪府 吹田市 江の木町３番１１号 第３ロンヂェビル Osaka (JP).

(81) 指定国 (国内): CN, JP, US.

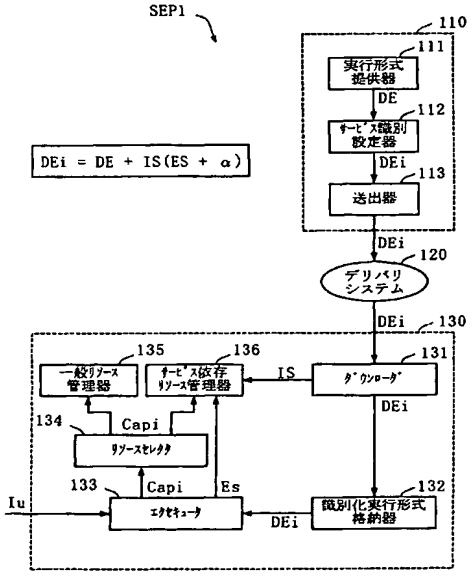
(84) 指定国 (広域): ヨーロッパ特許 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR).

添付公開書類:
— 国際調査報告書

[続葉有]

(54) Title: SERVICE SAFELY-EXTENSIBLE PLATFORM

(54) 発明の名称: サービス安全拡張プラットフォーム



111...EXECUTE FORM PROVIDING APPARATUS
112...SERVICE IDENTIFICATION SETTING APPARATUS
113...TRANSMISSION APPARATUS
120...DELIVERY SYSTEM
135...GENERAL RESOURCE MANAGEMENT APPARATUS
136...SERVICE-DEPENDENT RESOURCE MANAGEMENT APPARATUS
131...DOWNLOADER
134...RESOURCE SELECTOR
133...EXECUTOR
132...IDENTIFIABLE EXECUTE FORM STORAGE APPARATUS

(57) **Abstract:** A service extensible, completely new service safely-extensible platform that eliminates the need to deny that a processing that might cause a false access is included in an execute form and that can eliminate a false access by an execute form that realizes a digital content providing service. A service safely-extensible platform (SEP), in which the correspondence between a service (S) and an execute form (DE) is established and the service (S) is extended by changing or adding the execute form (DE), has a service-dependent API used for extending the service. The service is extended by the execute form (DE) only through a call of the service-dependent API.

(57) 要約: 実行形式に不正アクセスを引き起こす処理が含まれることを否定する必要無く、デジタルコンテンツの提供サービスを実現する実行形式による不正アクセスの排除できる、サービスの拡張可能な全く新しいサービス安全拡張プラットフォームを提供することを目的とする。サービス（Ｓ）と実行形式（ＤＥ）とが対応付けられており、前記実行形式（ＤＥ）の変更や追加によって前記サービス（Ｓ）の拡張が達成されるサービス安全拡張プラットフォーム（ＳＥＰ）は、サービスの拡張を行うサービス依存ＡＰＩを備え、かつ実行形式（ＤＥ）からのサービスの拡張はサービス依存ＡＰＩの呼び出しによってのみ行われる。

WO 03/107182 A1

２文字コード及び他の略語については、定期発行される各*PCT*ガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明細書

サービス安全拡張プラットフォーム

技術分野

デジタルコンテンツの提供サービスを実現する実行形式による不正アクセスの排除を、実行形式に不正アクセスを引き起こす処理が含まれることを否定する必要無く達成するという、新次元の安全性を実現した、サービスの拡張可能な全く新しい「サービス安全拡張プラットフォーム」に関する。詳述すれば、実行形式の配布元の確からしさからの推測を必要とせずに、他のサービスの状態変更、データ破壊、およびプラットフォーム自体のシステムダウンなどに代表される不正アクセスを排除できると共に、実行形式の変更や追加によってサービスの機能／諸元の変更や新規サービスの追加というサービス拡張も達成できるサービス安全拡張プラットフォームに関する。

背景技術

上述の如く、本発明が提供するものは、従来にない全く新しく提供する「サービス安全拡張プラットフォーム」であるので、従来の技術として示すべき適切な例を挙げるのは非常に困難である。それゆえに、先ず、本発明が初めて提唱すると信じるサービス安全拡張プラットフォームが固有に備える３つの主な特徴について以下に述べる。第１の特徴はサービスの拡張性であり、第２の特徴は安全性の確

保であり、そして第３の特徴はサービス拡張操作のシームレス化である。

第１の特徴である「サービスの拡張性」とは、サービス毎に異なるユーザインタフェースを持ち、ユーザインタフェースの変更や予期せぬサービスの追加が任意の時点で可能であることを言う。第２の特徴である「安全性の確保」とは、実行形式に不正アクセスを引き起こす処理が含まれることの否定を必要とせずに安全性が確保されることを言う。そして、第３の特徴である「サービス拡張操作のシームレス化」とは、サービス拡張のための操作手順が、サービスの利用時と同じ操作感のもとに行えることを言う。

上述のように、これら３つの主な特徴を全て満たす「サービス安全拡張プラットフォーム」を、従来技術において見いだすことはできない。しかしながら、強いて言えば、第１の特徴のみを満たすものとして、パーソナルコンピュータを利用してインターネットによるプッシュ型サービスを従来技術の一例として挙げることができる。ポイントキャストネットワーク社のポイントキャスト（Ｒ）は、ニュース配信サービスを実現したプッシュ型サービスである。パーソナルコンピュータを利用する複数のプッシュ型サービスの利用は、機械語で記述されたサービス毎に実装したブラウザをパーソナルコンピュータへのインストールすることに可能となる。

ここで、プッシュ型サービスにおける、本発明の第１の特徴である「サービスの拡張性」について説明する。プッシュ型サービスを実現するブラウザをパーソナルコンピュ

ータのハードディスクにインストールすることで新たなサービスを追加することができる。ブラウザは機械語で記述されたコンピュータプログラムであり、サービス毎に固有のそれぞれ異なるユーザインタフェースを提供する。

ブラウザのインストールは、従来、以下の様にして行われる。例えば、ブラウザをインストールする際には、先ず、ファイル転送プロトコル（例：ｆｔｐ）のクライアントを起動しておき、ブラウザをクライアントのハードディスクなどにダウンロードし、次にダウンロードしたブラウザを起動することで実現される。また、新たなブラウザを利用する前には、ブラウザの設定を、ブラウザに組み込まれた設定メニューで操作して実現する。

次に、本発明の第２の特徴である「安全性の確保」に関しては、従来は単なる憶測に基づくものであって、具体的な方策の裏付けを有するものではない。つまり、従来、ブラウザが安全であるということは、ブラウザが公式ホームページなど正式な配布先より入手したものであるので、悪意のある第３者がコンピュータウィルスなどを感染させてないはずだという単なる願望に過ぎない。

そして、サービスによる高度な提示機能を意味する本発明の第３の特徴である「サービス拡張操作のシームレス化」に関しては、従来の技術においては、サービス毎にブラウザを変更することで異なるユーザインタフェースを実現する。結果、操作のシームレス化の一部を構成するために必要な、複数サービスに対し異なるユーザインタフェースを持つ要件は満される。

図１８を参照して、上述のようなサービスの拡張は可能であるが安全ではない、従来のサービスプラットフォームについて具体的に説明する。同図に示すように、従来のサービスプラットフォームＰＦｃは、サービス提供源１６１０、デリバリシステム１２０、および端末１６３０とを含む。サービス提供源１６１０は、実行ファイル提供器１１１１およびファイル転送サーバ１６１１を含む。実行ファイル提供器１１１１は実行ファイルを格納したハードディスク装置でよい。また、ファイル転送サーバ１６１１はＷｅｂサーバでよい。なお、実行ファイルとは、図１９を参照して後述するように、端末１６３０のＯＳに直接渡して、実行されるプログラムファイルに代表されるバイナリデータである。

デリバリシステム１２０は、サービス提供源１６１０から送られる実行ファイルを空間的または時間的にはなれた端末１６３０に向けて伝送する。

端末１６３０は、エクセキュータ１１３３および実行ファイル格納器１１３２を含む。エクセキュータ１１３３は、実行ファイル格納器１１３２に格納された実行ファイルを起動することで端末１６３０におけるあらゆる処理を実行する。例えば、実行ファイルとしてファイル転送プログラムを起動した場合には、デリバリシステム１２０を経由して受信した、新たなサービスを実現する実行ファイルを実行ファイル格納器１１３２に格納させる。また、サービスを実現する実行ファイルを実行させれば、サービスをユーザに提供できる。

図１９に、サービスプラットフォームＰＦｃにおいて、実行ファイル格納器１１３２に格納されるデータを例示する。サービスＳ１の実行ファイルであるＦＥ（Ｓ１）およびサービスＳ２の実行ファイルであるＦＥ（Ｓ２）に並んで、実行ファイルの起動などのプラットフォーム操作を行うユーザインタフェースを提供するｓｈｅｌｌの実行ファイルＦＥ（ｓｈｅｌｌ）や、新たなサービスを実現する実行ファイルＦＥをデリバリシステム１２０経由で端末１６３０に導入するファイル転送プログラムであるＦＥ（ファイル転送）が実行ファイル格納器１１３２内に格納されている。これらの実行ファイルは何れも機械語で実装される。

図２０に、サービスプラットフォームＰＦｃのソフトウェア階層を示す。同図から読みとれるように、サービスプラットフォームＰＦｃにおいて、ファイル転送実行ファイルを起動することで、あらゆるサービスを実現する実行ファイルを新たに導入できる。このように、ＯＳ（Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ）が構成するＯＳ層の直上に、実行ファイルによってアプリケーション層が構成されている。結果、サービスを実現する実行ファイルが直接ＯＳのリソースを参照したり操作したりできる。それゆえに、ＯＳは、悪意を持った実行ファイルに対して無防備であるので、サービスプラットフォームＰＦｃはとうてい安全とは言えない。

また、本発明の第１の特徴である「サービスの拡張性」に関しては、新たなサービスの単純な導入は可能であるが

、本発明が提供するコンテンツの視聴とシームレス化した新たなサービスの導入については、従来は不可能である。つまり、コンテンツの視聴と新たなサービスの導入における操作性は、その見かけも手順も全く異なる。例えば、従来は、ブラウザをインストールする際には、ファイル転送プロトコルのクライアントを起動し、ブラウザをクライアントのハードディスクなどにダウンロードし、次にダウンロードしたブラウザを起動しなければならない。また、新たなブラウザを利用する前には、ブラウザの設定を、ブラウザによるコンテンツの視聴画面とは全く別の使い勝手の設定メニューを操作しなければならない。

また、従来は、全てのサービスに対して、共通のユーザーインタフェースを利用して機能拡張が行われる。そのために、サービス拡張するために、プログラムをインストールする際に必要な各種パラメータ（インストール先のディレクトリやユーザ名等）の入力が、全サービスで必要なパラメータの和集合となってしまう。そのために、特定のサービスの拡張プログラムをインストールするために、本来不要なパラメータの入力まで、ユーザに求めてしまうことがある。

一方、本発明にかかるサービス安全拡張プラットフォームが提供するサービスの柔軟な拡張性は、ブラウザを終了したり、コンテンツの視聴と全く異なる使い勝手の手順を踏むことなく、あくまでもコンテンツの視聴と同じ操作の使い勝手の中でサービスの拡張を行えることである。すなわち、サービスの拡張のためだけの特別な使い勝手の操作

をユーザに強要することなく、ユーザが最も慣れているであろうコンテンツの視聴と同じ使い勝手で、あたかも、通常の操作の延長として、サービスが拡張されていく状況を達成するものである。しかしながら、このように拡張性は、従来のサービスプラットフォームの提供できることではないことは上述の通りであり、そのために、全く新しいサービス拡張の実現方法が求められている。

次に、本発明の第２の特徴である「安全性の確保」に関しても、正式な配布先から入手したブラウザでもバグによって、他のプッシュ型サービスの動作に悪影響を与えたり、システム全体をダウンさせたりすることも少なくないのが現実である。そのため、従来においては、ブラウザのバイナリに正当性がある、すなわち、ブラウザのバイナリに不正アクセスの処理が含まれないであろうことを少なくとも保証する必要がある。しかし、その保証はあくまでも信頼関係に基づくものであって、ブラウザのバイナリに不正アクセス処理が含まれていないことを確実にするものではない。

一方、本発明にかかるサービス安全拡張プラットフォームが提供する安全性は、如何なる実行形式（従来例でのブラウザに対応）の実行によっても、他のサービスへの悪影響や、システム全体のダウンを起こさない、全く新しい次元の安全性である。当然、従来の技術においては、このように方式は存在せず、それゆえに全く新しい実現方法が求められているものである。

さらに、本発明の第３の特徴である高度な提示機能を実

現する「サービス拡張操作のシームレス化」に関しても、従来の技術ではサービス管理とコンテンツの視聴とのシームレス化は不可能である。その原因は、サービス管理はブラウザの設定メニューで行われていることにある。つまり、サービスの管理に関わることは、コンテンツの視聴画面では行えないのである。

一方、本発明にかかるサービス安全拡張プラットフォームが提供するサービスによる高度な提示機能は、コンテンツにサービスの設定を含めることで、コンテンツ視聴と同一の使い勝手の中で、コンテンツの持つ高度な表現を持ち、かつ、サーバから自由に提示内容を設定することを可能とする。

例えば、アンケートに記述しサーバに向けて回答を返信するといった、高度な記述能力を使用したコンテンツにサービスの設定機能を組み合わせることで、アンケートにユーザが興味にある内容を回答することで、自動的にコンテンツの取捨選択を行うためのユーザの嗜好情報が設定されると言った高度な提示機能を実現できる。また、ユーザにとっては、どこまでがコンテンツなのか、サービスの設定なのかを全く意識させることなく、高度な設定を簡単な操作で行える。

当然、従来の技術においては、本発明にかかるサービス安全拡張プラットフォームが提供するサービスによる高度な提示機能を実現する方式が存在せず、それゆえに全く新しい実現方法が求められているものである。

よって、本発明は、上述の３つの特徴を実現するサービ

ス安全拡張プラットフォームを提供することを目的とする。

発明の開示

本発明は、上記のような目的を達成するために、以下に述べるような特徴を有している。

第１の局面は、サービスと実行形式とが対応付けられており、実行形式の変更や追加によってサービスの拡張が達成されるサービス安全拡張プラットフォームであって、サービスの拡張を行うサービス依存ＡＰＩを具備し、かつ実行形式からのサービスの拡張はサービス依存ＡＰＩの呼び出しによってのみ行われることを特徴とする。

上述のように、第１の局面においては、実行形式の正当性に依らない安全性が確保できる。

第２の局面は、第１の局面において、サービスの拡張が新規サービスの新設であることを特徴とする。

第３の局面は、第１の局面において、サービスの拡張がサービス利用開始であることを特徴とする。

第４の局面は、第１の局面、第２の局面、および第３の局面の何れかにおいて、注目するサービスの拡張は、注目するサービスに対応付けられた実行形式からのサービス依存ＡＰＩの呼び出しによってのみ行われることを特徴とする。

第５の局面は、第１の局面、第２の局面、および第３の局面の何れかにおいて、複数のサービス間に親子関係が定義され、実行形式が要求するサービス依存ＡＰＩ呼び出し

がサービスに対応付けられるサービス依存リソースを処理対象として指定した際に、実行形式に対応付けられるサービスがサービスの先祖である場合にのみ、サービス依存リソースに対して処理可能であることを特徴とする。

第６の局面は、第１の局面、第２の局面、および第３の局面の何れかにおいて、メタサービスに対応付けられた実行形式がサービス依存ＡＰＩによって、サービスの少なくとも１つが拡張可能なことを特徴とする。

第７の局面は、第６の局面において、メタサービスに対応付けられた実行形式がサービス依存ＡＰＩによって、サービスの全てが拡張可能であり、メタサービスに対応付けられていない実行形式（ＤＥ）はサービス依存ＡＰＩによってサービスの拡張が不可能であることを特徴とする。

第８の局面は、第１の局面、第２の局面、および第３の局面の何れかにおいて、実行形式がコンテンツとして満たすべき条件を満たした制御コンテンツであり、制御コンテンツ（ＤＣ）がコンテンツの少なくとも１つと共にコンテンツとして伝送され、コンテンツ（ＤＣ）の少なくとも１つから制御コンテンツを指定する情報が伝送され、制御コンテンツによってのみサービス依存ＡＰＩの処理が可能であることを特徴とする。

第９の局面は、第８の局面において、サービス依存ＡＰＩによって、特定のサービスのコンテンツ（ＤＣ）の自動的な格納が制御されることを特徴とする。

第１０の局面は、第１の局面、第２の局面、第３の局面、第４の局面、第５の局面、第６の局面、第７の局面、第

８の局面、および第９の局面の何れかにおいて、実行形式を少なくとも１つのサービス提供部に送出し、実行形式を実行する少なくとも１つの端末で受信することを特徴とする。

なお、本局面の一実施の形態においては、ブラウザを終了したり、コンテンツの視聴と全く異なる使い勝手の手順を踏むことなく、あくまでコンテンツの視聴と同じ操作の使い勝手の中でサービスの拡張を行える。すなわち、サービスの拡張のためだけの特別の使い勝手の操作をユーザに強要することなく、ユーザが最も慣れているであろうコンテンツの視聴と同じ使い勝手のうえで、いわば、いつの間にかサービスが拡張されている状況を達成できる。

また、如何なる実行形式（従来例でのブラウザに対応）の実行によっても、他のサービスへの悪影響や、システム全体のダウンを起こさない、全く新しい次元の安全性を達成できる。

さらに、異なる実施の形態においては、コンテンツにサービスの設定を含めることで、コンテンツ視聴と同一の使い勝手の中で、コンテンツの持つ高度な表現を持ち、かつ、サーバから自由に提示内容を設定できる機能である。そして、ユーザにとっては、どこまでがコンテンツなのか、サービスの設定なのかを全く意識させることなく、高度な設定を簡単な操作で行える。

図面の簡単な説明

図１は、本発明の第１の実施の形態にかかる、サービス

安全拡張プラットフォームの構成を模式的に示すブロック図である。

図２は、図１に示したサービス安全拡張プラットフォームの識別化実行形式格納器に格納される識別化実行形式を示す説明図である。

図３は、図１に示したサービス安全拡張プラットフォームのサービス依存リソース管理器で管理されるサービス依存リソース管理テーブルの説明図である。

図４は、図１に示したサービス安全拡張プラットフォームのサービス依存リソース管理器による新サービスの登録管理動作を示すフローチャートである。

図５は、図１に示したサービス安全拡張プラットフォームの端末によるサービス実行動作を示すフローチャートである。

図６は、図１に示したサービス安全拡張プラットフォームの端末のソフトウェア階層を示す説明図である。

図７は、図１に示したサービス安全拡張プラットフォームによってユーザに提示される画面例を示す説明図である。

図８は、本発明の第２の実施の形態にかかるサービス安全拡張プラットフォームの構成を模式的に示すブロック図である。

図９は、図８に示したサービス安全拡張プラットフォームの識別化実行形式格納器に格納される識別化実行形式およびメタサービス実行形式を示す説明図である。

図１０は、図８に示したサービス安全拡張プラットフォ

ームの端末によるサービス実行動作を示すフローチャートである。

図１１は、本発明の第３の実施の形態にかかるサービス安全拡張プラットフォームの構成を模式的に示すブロック図である。

図１２は、図１１に示したサービス安全拡張プラットフォームの識別化実行形式格納器に格納される識別化実行形式および親子識別化実行形式を示す説明図である。

図１３は、図１１に示したサービス安全拡張プラットフォームの端末によるサービス実行動作を示すフローチャートである。

図１４は、本発明の第４の実施の形態にかかるサービス安全拡張プラットフォームの構成を模式的に示すブロック図である。

図１５は、図１４に示したサービス安全拡張プラットフォームのコンテンツ格納器に格納される制御コンテンツおよびコンテンツを示す説明図である。

図１６は、図１４に示したサービス安全拡張プラットフォームのサービス依存リソース管理器で管理されるサービス依存リソース管理テーブルの説明図である。

図１７は、図１４に示したサービス安全拡張プラットフォームの端末による制御コンテンツ実行動作を表すフローチャートである。

図１８は、従来の技術における、従来のサービスプラットフォームの構成を模式的示すブロック図である。

図１９は、図１８に示したサービスプラットフォームの

実行ファイル格納器に格納される実行ファイルを示す説明図である。

図２０は、図１８に示したサービスプラットフォームの端末におけるのソフトウェア階層を示す説明図である。

発明を実施するための最良の形態

先ず、本発明にかかるサービス安全拡張プラットフォームの基本的概念について説明する。本発明にかかるサービス安全拡張プラットフォームにおいて、サービスは、従来のようなＯＳによって直接実行される、例えばバイナリの実行ファイルではなく、ＯＳに対する実行を指示するエクセキュータにより解釈されるいわばスクリプトとして構成される実行形式として構成される。そして、実行形式は、ＡＰＩ呼び出しを含み、所定のサービスに対応付けられる。

ＡＰＩは、呼び出し時に、対応付けられたサービスに依存した挙動をするサービス依存である。サービス依存ＡＰＩは、含まれる実行形式に対応付けられたサービスに固有のリソースに対して処理する。また、実行形式からリソースへのあらゆる処理はＡＰＩ経由に限定され、かつ、サービス依存ＡＰＩにより処理されるリソースに対して処理を行うサービス依存ＡＰＩ以外のＡＰＩは存在しない。また、実行形式内に記述するＡＰＩのセットは、全てのサービスに対し共通に予め用意されたものであり、同一である。

このように構成することによって、従来における実行ファイル（あるいは本発明における実行形式）の正当性に依

らない新次元の安全性を確保し、サービスの拡張が可能な全く新しい「サービス安全拡張プラットフォーム」を創出する。同プラットフォームにおいては、サービスを実現する実行形式の一例であるブラウザを動作させることで、サービスのインスタンスであるコンテンツが視聴できる。コンテンツの通常の視聴は、コンテンツに含まれるＡＰＩを逐次的に呼び出して実現される。そのために、実行形式（ブラウザ）は、ＡＰＩの一種であるサービス依存ＡＰＩを含んで構成される。つまり、コンテンツの視聴の最中に、サービス依存ＡＰＩが呼び出されることで、サービスが拡張される。結果、本発明においては、サービスの拡張は、ユーザにはコンテンツの視聴操作の一つとして見える。

サービスを拡張するユーザインタフェースは、サービス依存ＡＰＩを含む実行形式によって決定される。実行形式をサービスに対応付けることで、個々のサービスのそれぞれに適したユーザインタフェースを利用してサービスが拡張できる。

サービス拡張のユーザインタフェースを、サービス毎に最適に構成できる。例えば、プッシュ型サービス／蓄積型放送サービスの場合、同サービスの会員の中で、特定の会員のみ加入可能なプレミアムサービスの契約を、通常のサービスのコンテンツを利用してそのような特定の会員のみに対して勧誘できる。つまり、目的などに応じてユーザインタフェースを必要十分にカスタマイズできるので、より適切且つ強力にサービスの勧誘が出来る。

携帯電話等で利用されるＪａｖａアプリ（ｉアプリ等）

に本発明を適応すれば、ユーザが購読しているサービスを実現するアプリケーションソフトウェアに、サービスの機能を拡張するプログラムを記述することで、同サービスの機能拡張を実現できる。例えば、スキンなどのｉアプリの機能をプラグイン的に拡張する操作をｉアプリ自身で実現できる。

更には、コンテンツ内に含まれるスクリプトに条件判断として、問いかけに応答して受信機の型番やメーカの種別を表す文字列を返すＡＰＩを呼び出すようにコンテンツを記述することで、受信機の型番やメーカ毎にユーザインタフェースを変化させることができる。

また、デジタル放送におけるペイパービュー番組に適応すれば、配信コンテンツに適切なアルゴリズム記述することで、おためし無料視聴が可能な期間やチャンネルをコンテンツにプログラミングしたアルゴリズムによって管理できる。デジタル放送でのペイパービュー番組の場合、ペイパービューのチャンネルのデータ放送を通してペイパービューの契約ができる。また、データ放送のゲームで高得点をあげたユーザに対してのみ安価な料金でサービス契約するというような、契約促進に繋がる、ユーザ毎に異なるサービスの提供が可能である。

次に、コンテンツに、サービス依存ＡＰＩが実行されるようにプログラミングすることによって、コンテンツの記述能力を損なうことなく、ユーザがコンテンツに対して所定の操作を行うことによって、コンテンツに関してシームレスにサービス拡張ができる。

例えば、Ｗｅｂブラウザのｐｌｕｇ－ｉｎをインストールする場合には、同ｐｌｕｇ－ｉｎのインストールプログラムを入手する為に、同ｐｌｕｇ－ｉｎ開発ベンダーのダウンロードページにアクセスせずに、コンテンツの受信時にＰｌｕｇ－ｉｎをインストール出来る。また、ＷｅｂブラウザのＰｌｕｇ－ｉｎの商品名について、ユーザが熟知しておく必要はなく、「商品を３６０度の角度からみた動画」というような、コンテンツ本意の説明を選択するだけで実行できる。

また、電子ブックプレーヤやＰＤＡに適応すれば、特定ホームページに入った時に同ホームページで必要とされる掲示板表示専用ブラウザなどが、ユーザが気づかないうちに自動的に使えるように準備される機能を提供できる。いわば、コンテンツの視聴とサービス拡張に用いられるユーザインタフェースをシームレスに構成出来る。コンテンツ操作に於けるのと同じ操作性のユーザインターフェースを利用してサービス拡張ができるので、ユーザにかかる負担が小さい。

次に、逐次的に実行されるＡＰＩとして、サービス依存ＡＰＩを追加することによって、実行形式を実行させれば、サービス依存ＡＰＩも他のＡＰＩ同様逐次的に実行される。結果、自動でサービスの拡張を実行するように実行形式をプログラミングすることも可能である。また、完全に自動化することで、旅行中の専用サービスや無料試用など、その時限りのサービス拡張もユーザが無意識のうちに実現できる。

例えば、電子ブックプレーヤーやＰＤＡの場合、ページめくりの際の画面効果をタイトルに付随し、そのタイトルでしか使えないプレミアムなプラグインとして提供できる。また、成田空港内のバーチャル施設案内など、その場でしか意味のないサービスを自動で利用できる。

次に、本発明においては、ブラウザを終了したり、コンテンツの視聴と全く異なる使い勝手の手順を踏むことなく、あくまでもコンテンツの視聴と同じ操作の使い勝手の中でサービスの拡張が行えることを保証する。つまり、サービスの拡張のためだけに用意される特別の使い勝手の操作をユーザに強要することなく、ユーザが最も慣れているであろうコンテンツの視聴と同じ使い勝手で、ユーザがそれと特に意識することなく、気が付けばサービスが拡張されている環境を提供する。

例えば、プッシュ型サービス／蓄積型放送サービスの場合、ビューワのバージョンアップならそのビューワ自体のアップグレード機能、別のサービスの導入ならＷｅｂブラウザを起動してダウンロードといった複数のユーザインタフェースの作法に慣れている必要がない。

つまり、コンテンツの内容や、ユーザの好みに応じて、複数のブラウザを使い分ける必要のある場合にも、ユーザは様々なユーザインタフェースの異なる操作性に慣れる必要がない。それ故に、不慣れなユーザでも安心し、且つ容易に、機能拡張できる。そして、視聴装置の出荷後に、ユーザインタフェースが固定化されることなく、ユーザの使い勝手の応じて、その操作性を改善出来る。

更には、全てに共通するが、如何なる実行形式（従来例でのブラウザに対応）の実行によっても、他のサービスへの悪影響や、システム全体のダウンを起こさない、全く新しい次元の安全性が確保できる。

（第１の実施の形態）

図１、図２、図３、図４、図５、図６および図７を参照して、本発明の第１の実施の形態にかかるサービス安全拡張プラットフォームについて説明する。図１に示すように、本実施の形態にかかるサービス安全拡張プラットフォームＳＥＰ１は、サービス提供源１１０、デリバリシステム１２０、および端末１３０を含む。

サービス提供源１１０は、端末１３０が実行すべき新たなサービスを実現するための実行形式を送出する。デリバリシステム１２０は、サービス提供源１１０が送出する情報を端末１３０に向けて時間的かつ／あるいは空間的に移動させる。デリバリシステム１２０は、インターネット通信網、放送や通信の無線ネットワーク、あるいは、ＤＶＤ－ＲＯＭ（Ｄｉｇｉｔａｌ　Ｖｅｒｓａｔｉｌｅ　Ｄｉｓｋ－Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）などのパッケージメディアと物理的な物流システムの組み合わせで構成できる。

端末１３０は、デリバリシステム１２０経由で受け取った情報を用いて、サービス提供源１１０から提供されたサービスを実行して、実行結果をユーザに提供する。なお、図１においては、簡便化のために、サービス提供源１１０、デリバリシステム１２０、および端末１３０のそれぞれ

の台数比が１：１：１であるように表されている。しかしながら、台数比は放送の形態に類似した１：１：ｃ（ｃは任意の自然数）や、インターネットなどのａ：１：ｃ（ａは任意の自然数）や、あるいは複数のデリバリシステムを持つ場合のａ：ｂ：ｃ（ｂは任意の自然数）であってもよい。このような、一般化は以下に述べる全ての実施の形態において当てはまる。

サービス提供源１１０は、実行形式提供器１１１、サービス識別設定器１１２、および送出器１１３を含む。実行形式提供器１１１は、サービスを実現する実行形式を格納し、必要に応じて出力する。この実行形式は、インターネットで用いられるＨＴＭＬ言語（ＨｙｐｅｒＴｅｘｔ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）や、日本のデジタル放送のデータ放送で用いられるＢＭＬ言語（Ｂｒｏａｄｃａｓｔ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）などの、ＳＧＭＬ（Ｓｔａｎｄａｒｄ Ｇｅｎｅｒａｌｉｚｅｄ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）／ＸＭＬ（ｅＸｔｅｎｓｉｂｌｅ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）系のマークアップ言語や、仮想マシン上で動作するＪａｖａ（Ｒ）言語、あるいは機械語などでよい。ただし、実行形式は、後述する端末１３０に備えられたエクセキュータを介して、端末１３０のＯＳに対して渡されて実行されるように構成される。

サービス識別設定器１１２は、実行形式提供器１１１の出力する実行形式ＤＥに対して、実行形式ＤＥの属性等の副次的な事項を表す付随情報ＩＳを生成すると共に実行形

式ＤＥに付与して識別化実行形式ＤＥｉを生成する。付随情報ＩＳは、対応する実行形式ＤＥが実現するサービスとを対応付ける情報であるサービス識別情報Ｅｓと、その実行形式ＤＥの使用条件、使用状態、および対応するサービス内容などの情報を表す副次情報αを含む。サービス識別情報Ｅｓは、例えば、重複しない数字であるＩＤや名称などでよい。副次情報αはテキストでもコードでもよい。また、識別化実行形式ＤＥｉは、実行形式ＤＥと付随情報ＩＳを一体的に生成してもよいし、互いに独立して生成してもよい。本実施の形態においては、視認性を考慮して付随情報ＩＳは独立して生成される場合を例に説明する。

送出器１１３は、サービス識別設定器１１２から入力される識別化実行形式ＤＥｉ（実行形式ＤＥおよび付随情報ＩＳ）をデリバリシステム１２０に送出する。この送出を実現する伝送モデルは、いわゆるプル型およびプッシュ型の何れでもよい。

プル型とは、インターネットのホームページ閲覧に用いられる伝送プロトコルであるＨＴＴＰ（ＨｙｐｅｒＴｅｘｔ Ｔｒａｎｓｐｏｒｔ Ｐｒｏｔｏｃｏｌ）などで見られるように、受信側である端末１３０からの要求（ｄｅｍａｎｄ）に基づき送出する伝送モデルである。またプッシュ型とは、デジタル放送の伝送に用いられる伝送プロトコルであるＤＳＭ－ＣＣ（Ｄｉｇｉｔａｌ Ｓｔｏｒａｇｅ Ｍｅｄｉａ Ｃｏｍｍａｎｄ ＆ Ｃｏｎｔｒｏｌ）データカルーセルなどで見られるように、受信側の要求に関わらず所定のタイミングで送出側から送出する伝送モデル

である。

端末１３０は、ダウンローダ１３１、識別化実行形式格納器１３２、エクセキュータ１３３、リソースセレクタ１３４、一般リソース管理器１３５、およびサービス依存リソース管理器１３６を含む。ダウンローダ１３１は、デリバリシステム１２０から伝送されてくる、識別化実行形式ＤＥｉ（実行形式ＤＥおよび付随情報ＩＳ）を受信し、受信した識別化実行形式ＤＥｉを識別化実行形式格納器１３２に書き込むと共に、識別化実行形式ＤＥｉから付随情報ＩＳを抽出してサービス依存リソース管理器１３６に出力する。付随情報ＩＳは、上述のようにサービス識別設定器１１２で設定されたサービス識別情報Ｅｓを含む。ダウンローダ１３１は、送出器１１３とデリバリシステム１２０で実現される伝送モデルに整合しており、プッシュ型でもプル型でも実施可能である。

識別化実行形式格納器１３２は、ダウンローダ１３１により書き込まれた識別化実行形式ＤＥｉ（実行形式ＤＥおよび付随情報ＩＳ）を格納する。また、識別化実行形式格納器１３２は、要求に応じて格納した識別化実行形式ＤＥｉをエクセキュータ１３３に出力する。識別化実行形式格納器１３２はハードディスクドライブ（ＨＤＤ）やＤＶＤ－ＲＡＭなどの記録媒体や、フラッシュメモリやＲＡＭなどの半導体メモリを用いて構成できる。

図２に、本実施の形態において、識別化実行形式ＤＥｉが識別化実行形式格納器１３２に格納される様子を示す。同例において、サービスＳ１、サービスＳ２乃至サービス

Ｓｎ（ｎは任意の自然数）に関して、識別化実行形式ＤＥｉ（Ｓ１）はサービスＳ１に対応し、識別化実行形式ＤＥｉ（Ｓ２）はサービスＳ２に対応し、識別化実行形式ＤＥｉ（Ｓｎ）はサービスＳｎに対応する。

そして、識別化実行形式ＤＥｉ（Ｓ１）は、それぞれサービスＳ１に対応するサービス識別情報Ｅｓ（Ｓ１）と副次情報α１から成る付随情報ＩＳ（Ｓ１）と実行形式ＤＥを含む。具体的には、サービス識別情報Ｅｓ（Ｓ１）が実行形式ＤＥとサービスＳ１とを対応付けている。同様に、識別化実行形式ＤＥｉ（Ｓ２）は、それぞれサービスＳ２に対応するサービス識別情報Ｅｓ（Ｓ２）と副次情報α２から成る付随情報ＩＳ（Ｓ２）と実行形式ＤＥを含む。さらに、識別化実行形式ＤＥｉ（Ｓｎ）は、サービスＳｎに対応するサービス識別情報Ｅｓ（Ｓｎ）と副次情報αｎから成る付随情報ＩＳ（Ｓｎ）と実行形式ＤＥを含む。実行形式ＤＥはサービス識別情報Ｅｓ（Ｓｎ）によってサービスＳ２に対応付けられている。

サービスを個々に区別する必要のない場合、識別化実行形式ＤＥｉはサービス識別情報Ｅｓおよび副次情報αから成る付随情報ＩＳと実行形式ＤＥで構成されると表現する。そして、サービスを個々に区別する必要のある場合は、識別化実行形式ＤＥｉ（Ｓｏ）は、サービス識別情報Ｅｓ（Ｓｏ）および副次情報αｏから成る付随情報ＩＳ（Ｓｏ）と実行形式ＤＥで構成されると表現する。なお、ｏはｎ以下の任意の自然数である。

図１に戻って、エクセキュータ１３３は、識別化実行形

式格納器１３２から入力される識別化実行形式ＤＥｉに含まれる実行形式ＤＥを解釈してサービスを実行する。ただし、識別化実行形式格納器１３２に格納された識別化実行形式ＤＥｉに含まれる実行形式ＤＥは、エクセキュータ１３３以外では端末１３０内で解釈実行されることはない特徴を有していることは上述の通りである。エクセキュータ１３３は、また、ユーザからのキーボードやポインティングデバイスや音声入力デバイスといった入力デバイスからの入力Ｉｕや、ＧＵＩの画面表示や音声出力などの出力デバイスへの出力、といった対話処理を実行形式ＤＥに基づいた手順で実現する。

エクセキュータ１３３は、実行形式ＤＥがＪａｖａ（Ｒ）言語やマークアップ言語であれば仮想マシンや、一般にはブラウザと呼ばれる実行環境でよい。また、実行形式ＤＥが機械語であれば、それを実行させるためのＯＳに付属するライブラリなどのミドルウェア群などである。

エクセキュータ１３３は、実行形式ＤＥを解釈実行していく際に、実行形式ＤＥにＡＰＩ（Ａｐｐｌｉｃａｔｉｏｎ　Ｐｒｏｇｒａｍ　Ｉｎｔｅｒｆａｃｅ）呼び出しＣａｐｉが含まれている場合には、リソースセレクタ１３４を経由して一般リソース管理器１３５もしくはサービス依存リソース管理器136に対してＡＰＩ呼出Ｃａｐｉを発行して、処理の実行を要求する。また、エクセキュータ１３３は、現在実行している実行形式ＤＥに対応付けられたサービス識別情報Ｅｓをサービス依存リソース管理器１３６に通知する。

リソースセレクタ１３４は、エクセキュータ１３３から発行されたＡＰＩ呼出Ｃａｐｉに基づいて、一般リソース管理器１３５およびサービス依存リソース管理器１３６の何れに対して、ＡＰＩ処理が要求されているのかを判断する。そして、要求されていると判断されている方に、ＡＰＩ呼出Ｃａｐｉを伝達する。なお、ＡＰＩ呼出Ｃａｐｉには、一般リソース管理器１３５が処理すべき一般のＡＰＩ（以後、「一般ＡＰＩ」と呼ぶ）と、サービス依存リソース管理器１３６が処理すべきＡＰＩ（以降、「サービス依存ＡＰＩ」）とに分類される。

リソースとは、端末１３０がエクセキュータ１３３を経由して、参照、変更、あるいは制御可能なあらゆる計算機資源を言う。すなわち、ＲＡＭへの１次記録器やＨＤＤなどの２次記録器に格納されるデータ構造、制御可能な入出力デバイスに対するアクセス権と具体的な入出力制御、および通信制御などが含まれる。これらリソースのうち、他のサービスへの影響を与えることなく処理可能なリソースを一般リソースとし、サービス毎に存在し、かつサービスの拡張の際に処理が必要なリソースをサービス依存リソースと定義する。そして、一般リソースに対するＡＰＩを一般ＡＰＩとし、サービス依存リソースに対するＡＰＩをサービス依存ＡＰＩと定義する。

さらに、２つのＡＰＩと同一のリソースとの関係に例をおいて説明する。そして、一方のＡＰＩはリソースを参照して他のサービスに影響しないが、他方のＡＰＩはリソースを変更して他のサービスに影響を及ぼすと想定する。こ

の場合、前者が一般ＡＰＩに対応し、後者がサービス依存ＡＰＩに対応する。しかしながら、説明の簡便化のために、参照のためのリソースと変更のためのリソースとがそれぞれ独立に存在すると見なして、前者と後者の内容が整合されていると捉えて説明する。

サービス依存ＡＰＩの具体例としては、サービスの利用状態を変更する関数（以後、「サービス利用状態操作関数」と称する）などが考えられる。サービス利用状態操作関数によって、利用者が端末１３０で個別のサービスを利用するのかしないのかを指定し、サービスを提供する際に端末１３０が行うべき処理の起動と停止などを制御する。具体的には実行形式ＤＥあるいは識別化実行形式ＤＥｉの内部データの初期化や必要な情報の受信などの前処理や、デリバリシステムを経由したサービス提供部に対するサービス利用契約の締結、課金、およびユーザ登録などである。

一方、一般ＡＰＩの具体例としては、端末受信部１３０の画面表示やキーボード入力といった入出力デバイスへの操作や、ＲＡＭへの一時記憶やＨＤＤへの２次記憶に対するデータの読み書きなどがある。ただし、一般ＡＰＩのリソースである画面表示を例にした場合に、複数のサービスに対応する画面表示が同時に出現する際には、サービス間で競合が発生することも考えられる。しかしながら、本実施の形態においては、実際には同時には１つのサービスのみが画面を占有するという制約を一般リソース管理器１３５やＡＰＩの呼び出し方法などで実現すれば、あらゆる実行形式ＤＥに対しても実際には競合が発生しない。

一般リソース管理器１３５は、一般リソースを格納および管理する。一般リソース管理器１３５は、さらに、リソースセレクタ１３４から入力されるＡＰＩ呼出Ｃａｐｉに基づいて、一般リソースへの参照や操作などを行う。例えば、画面描画のＡＰＩが呼び出されると画面描画行う一般リソースであるグラフィック表示デバイスに対して命令を発する。

サービス依存リソース管理器１３６は、サービス依存リソースを格納および管理する。リソースセレクタ１３４から入力されるＡＰＩ呼出Ｃａｐｉに基づいて、サービス依存リソースＲＳへの参照や操作などを行う。また、ダウンローダ１３１から入力される付随情報ＩＳに基づいて、サービス依存リソースＲＳの管理処理を行う。

図３を参照して、サービス依存リソース管理器１３６に格納されるサービス依存リソースＲＳを管理するために生成されるサービス依存リソース管理テーブルＴｒｓについて説明する。同図に例示するように、サービス依存リソース管理テーブルＴｒｓは、少なくとも、ｎ種類のサービスＳ１～Ｓｎを表す複数の行Ｌ１～Ｌｎと、サービスＳ毎の利用状態を表す２列Ｃ１およびＣ２からマトリックス状に構成されるデータベースである。具体的には、同図において行ＬＩがサービスＳ１に対応し、行Ｌ２がサービスＳ２に対応し、行Ｌｎがサービスｓｎ（この場合、ｎは３以上の自然数）に対応している。そして、列Ｃ１はサービス識別情報Ｅｓに対応し、列Ｃ２は利用状態に対応する。

なお、より具体的に言えば、列Ｃ１および列Ｃ２の値は

、それぞれ図２に模式的に表したように、サービス識別情報Ｅｓ（Ｓｎ）のＳｎおよび副次情報αｎに基づいて決定されて書き込まれる。図３に示す例においては、行Ｌ１に示されるサービスＳ１は「未利用」であり、行Ｌ２に示されるサービスＳ２は「利用」状態であり、行Ｌｎに示されるサービスＳｎは「利用」状態であることが分かる。このように行Ｌ１～Ｌｎのそれぞれは、異なるサービスＳ１～Ｓｎのサービスを識別する情報を蓄えるために設けられている。この意味において、行Ｌ１～Ｌｎ（行Ｌｏ）をサービス識別行と呼ぶ。

次に、新たなサービス識別行を追加する際の動作について説明する。ダウンローダ１３１は、サービス（識別化実行形式ＤＥｉ）を受け取ると、識別化実行形式格納器１３２に格納されている実行形式ＤＥが対応するサービス依存リソースＲＳを規定するサービス識別行がサービス依存リソース管理テーブルＴｒｓに追加される。これらの処理によって、自動的にサービスが端末１３０に導入されることによって、新サービスがサービス依存リソース管理器１３６で登録・管理される。

以下に、図４を参照して、サービス依存リソース管理器１３６による新サービスの登録管理ルーチンの動作について説明する。新たなサービスに対応する識別化実行形式ＤＥｉ（実行形式ＤＥおよび付随情報ＩＳ）は、サービス提供源１１０から端末１３０に送出されて、先ずダウンローダ１３１に入力される。ダウンローダ１３１は、識別化実行形式ＤＥｉから付随情報ＩＳを抽出して、サービス依存

リソース管理器１３６に出力する。そして、サービス依存リソース管理器１３６においては、付随情報ＩＳが入力されて時点で、本ルーチンの動作が開始される。

そのため、ステップＳ５０２において、付随情報ＩＳがサービス依存リソース管理器１３６に入力されているか否かが判断される。入力されていない場合、Ｎｏと判断されて、本ステップにおける処理が繰り返される。一方Ｙｅｓの場合、処理は次のステップＳ５０４に進む。つまり、サービス依存リソース管理器１３６は、ダウンローダ１３１から付随情報ＩＳを受け取るまではサインサービスの登録管理処理は、実質的に開始されない。

ステップＳ５０４において、ステップＳ５０２で受け取った付随情報ＩＳのサービス識別情報Ｅｓが示すサービスＳが、サービス依存リソース管理器１３６に格納されているサービス依存リソース管理テーブルＴｒｓに既に登録されているか否かが判断される。含まれない、つまり、未登録サービスである場合は、Ｎｏと判断されて、処理は次のステップＳ５０６に進む。

ステップＳ５０６において、サービス依存リソース管理器１３６において、サービス依存リソース管理テーブルＴｒｓに新サービスに対応する新サービス識別行（Ｌｎ＋１）が追記される。以降、図３に示すサービス依存リソース管理テーブルＴｒｓを例として説明する。サービス依存リソース管理テーブルＴｒｓには、既にサービスＳ１～Ｓｎまでが登録されているので、新たなサービスＳ（ｎ＋１）を登録するために、サービス識別行Ｌ（ｎ＋１）が追加さ

れる。そして、処理は次のステップＳ５０８に生成される。

ステップＳ５０８において、新たに受領した付随情報ＩＳに含まれるサービス識別情報Ｅｓに基づいて、行Ｌ（ｎ＋１）列Ｃ１にサービスＳ（ｎ＋１）を識別する値であるＳ（ｎ＋１）が記入される。そして、処理は次のステップＳ５１０に進む。

ステップＳ５１０において、サービス依存リソース管理テーブルＴｒｓに追加された行Ｌ（ｎ＋１）列Ｃ２の利用状態の欄に「未利用」が記入される。これは、新規サービス（識別化実行形式ＤＥｉ）が受信された際の初期状態としては、「未利用」を設定するようにしているからである。しかしながら、初期状態が「利用」であったり、一定期間の間デモとして試用するなどの設定にしてもよいし、また、これらの初期状態の何れを採るか示す情報を、付随情報ＩＳ（副次情報α）としてサービス識別設定器１１２によってサービス毎に与えてもよい。本ステップの処理の終了後、本ルーチンを終了する。

一方、上述のステップＳ５０４において、Ｙｅｓ、つまり含まれると判定された場合、上述のステップＳ５０６、ステップＳ５０８およびステップＳ５１０をスキップして、本ルーチンにおける処理を終了する。すなわち、ダウンローダ１３１が受け取ったサービスが既に導入されているサービスである場合には、サービス登録は不要であるので、本ルーチンの処理が直ちに終了される。

次に、図５を参照して、端末１３０によるサービス実行

ルーチンの動作について説明する。具体的には、端末１３０において、エクセキュータ１３３が呼び出すＡＰＩに対する実行形式ＤＥが実行されることによって、サービス実行が実現される。つまり、エクセキュータ１３３が識別化実行形式格納器１３２から入力される識別化実行形式ＤＥｉに含まれる実行形式ＤＥを実行させるために、ＡＰＩ呼出Ｃａｐｉを発行した時点で、本ルーチンにおける実質的処理が開始される。

よって、ステップＳ５１２において、リソースセレクタ１３４によって、エクセキュータ１３３から発行されたＡＰＩ呼出Ｃａｐｉに基づいて、呼び出されたＡＰＩがサービス依存ＡＰＩであるか否かが判断される。サービス依存ＡＰＩであれば、Ｙｅｓと判断されて、処理は次のステップＳ５１４に進む。

ステップＳ５１４において、サービス依存リソース管理器１３６によって、現在実行している実行形式ＤＥに対応付けられたサービスのサービス識別情報Ｅｓを、エクセキュータ１３３から得る。そして、処理は、次のステップＳ５１６に進む。

ステップＳ５１６において、サービス依存リソース管理器１３６によって、ＡＰＩ呼出Ｃａｐｉが処理対象として指定するサービス依存リソースＲＳが、ステップＳ５１４で検出された現在実行中の実行形式ＤＥに対応するサービスに対応するか否かが判断される。実行中の実行形式ＤＥに対応するサービスに対応する場合は、Ｙｅｓと判断されて処理は次のステップＳ５１８に進む。

ステップＳ５１８において、サービス依存リソース管理器１３６によって、サービス依存リソースに対するサービス依存ＡＰＩの処理が行なわれる。そして、本ルーチンにおける処理は終了される。

一方、上述のステップＳ５１２において、Ｎｏ、つまりサービス依存ＡＰＩではない（すなわち一般ＡＰＩである）と判断される場合、処理はステップＳ５２０に進む。

ステップＳ５２０において、一般リソース管理器１３５によって、一般リソースに対して処理が行われる。そして、本ルーチンにおける処理は終了される。

さらに、上述のステップＳ５１６においてＮｏ、つまり実行中の実行形式ＤＥに対応するサービスに対応しない場合、処理はステップＳ５２１に進む。

ステップＳ５２１において、エラー処理が行われた後に、本ルーチンが終了される。このように、実行中の実行形式ＤＥが対応しているサービス（Ｓ５１２でＹｅｓ）であっても、操作不可能（Ｓ５１６でＮｏ）なサービス依存リソースに対するＡＰＩ処理を許可しないように設定している。

言い換えれば、本実施の形態においては、実行中の実行形式ＤＥに対応付けられたサービスに関するサービス依存リソースのみが操作可能と設定される。このため、実行形式ＤＥによって如何なるＡＰＩ呼び出しを発行させても、サービス依存リソース管理器１３６は他のサービスに対する参照や操作を排除できる。つまり、他のサービスに対する参照や操作を行うようなサービスの実行形式ＤＥが入力

されても、ステップＳ５１６を経てステップＳ５２１でエラー処理が行われて、そのような参照や操作を防止すると共に、そのような要求を検出できる。

上述のように、本実施の形態においては、ステップＳ５１２において、リソースセレクタ１３４によって、サービス依存リソースに対する処理はサービス依存ＡＰＩでのみ操作できる様に規制されている。これについて、図６に示すサービス安全拡張プラットフォームＳＥＰ１の端末１３０のソフトウェア階層を参照して説明する。

図６に示すように、サービス安全拡張プラットフォームＳＥＰ１の端末１３０をソフトウエア構成から見れば、最下層に基本ソフトウェアであるＯＳにより実現されるＯＳ層が存在する。そして、ＯＳ層の直上に、それぞれ一般リソース管理器１３５、サービス依存リソース管理器１３６、および識別化実行形式格納器１３２を機能させる一般リソース管理ソフトウェア、サービス依存リソース管理ソフトウェア、および識別化実行形式格納ソフトウェアを有する。そして、一般リソース管理ソフトウェアおよびサービス依存リソース管理ソフトウェアの直上にはリソースセレクタ１３４を機能させるリソースセレクトソフトウェアを有する。これらの、一般リソース管理ソフトウェア、サービス依存リソース管理ソフトウェア、リソースセレクトソフトウェア、および識別化実行形式格納ソフトウェアは、ミドルウェア層を構成する。

ミドルウェア層のリソースセレクト層の直上には、エクセキュータ１３３を機能させるエクセキュートソフトウェ

ア（エクセキュータ）が位置し、識別化実行形式格納ソフトウェアの直上にはダウンローダ１３１を機能させるダウンロードソフトウェア（ダウンローダ）が位置している。そして、これらのエクセキュートソフトウェアとダウンロードソフトウェアは共に、アプリケーション層を構成している。

そして、アプリケーション層のエクセキュートソフトウェアの直上には、各サービスＳ１～Ｓｎを実行するサービスＳ１～Ｓｎ実行形式が位置して、コンテンツ層を構成している。

このように、図６に示されるソフトウエア構成から明らかなように、サービス安全拡張プラットフォームＳＥＰ１においては、サービスの実行形式ＤＥはリソースセレクタ経由のＡＰＩ呼び出したリソースを参照、あるいは操作したりできない。よって、サービス依存リソースに対しては、サービス依存ＡＰＩを呼び出すことが必須であるので、サービス依存リソース管理器１３６を経由しなければ参照したり操作したりできない。

次に、図７に示す、サービス安全拡張プラットフォームＳＥＰ１によってユーザに提示される画面例を参照して、新たに追加されたサービスを実際に利用する状態に変更する際の動作について簡単に説明する。図７には、実行形式ＤＥにより表示される新規サービス利用開始の是非をユーザに問い合わる画面の一例が示されている。

画面ＳＭは、新たなサービスである「マイ・ニュース・サービス」を実現する実行形式ＤＥをエクセキュータ１３

３で実行することで提示される画面の１例である。画面ＳＭ上にはサービスの利用を宣言するボタンＢＹと、利用しないことを宣言するボタンＢＮが配置されている。ユーザは、入力デバイスを操作してボタンＢＹを選択するとこのサービスの利用が開始される。

ここでボタンＢＹが選択された場合の動作について説明する。ボタンＢＹには、サービスの利用を開始を宣言するサービス依存ＡＰＩを起動する様に実行形式ＤＥ中にプログラミングされている。このためボタンＢＹが選択されるとリソースセレクタ１３４を経由してサービス依存ＡＰＩがサービス依存リソース管理器１３６に届き、サービス依存リソース管理テーブルＴｒｓに記載されているサービス依存リソースＲＳに該当するサービスの利用状態の欄の値を「利用」に書き換える。

なお、図７に示す例では、サービスの単純な利用開始について述べたが、サービス提供部に対するサービス利用契約の締結やユーザ登録、あるいは蓄積型のデータ放送における視聴前の事前の自動蓄積処理開始なども同様に実施可能である。

このように、サービスの使用／未使用の状態遷移など、サービスに関する操作をサービス自身の実行形式ＤＥによってユーザとの対話を行うことが可能となる。さらに、如何なる悪意を持った実行形式ＤＥを他のサービスが実装された場合であっても、サービスやプラットフォームに対する、誤動作やハングアップに代表される、あらゆる悪影響を排除できる。

上述のように、本発明の第１の実施の形態にかかるサービス安全拡張プラットフォームＳＥＰ１においては、実行形式ＤＥに不正な処理を引き起こすコードが含まれていないことを実行形式ＤＥの配布元の身元の確からしさから類推するような不確実な安全性ではなく、どの様な実行形式ＤＥを想定しても他のサービスやプラットフォーム自体に対する不正な処理の影響を引き起こさない、完全な安全性を、サービスの拡張性を保ったままで確保できる。

（第２の実施の形態）

以下に、図８、図９、および図１１を参照して、本発明の第２の実施の形態にかかるサービス安全拡張プラットフォームについて説明する。図８に示すように、本実施の形態にかかるサービス安全拡張プラットフォームＳＥＰ２は、図１に示すサービス安全拡張プラットフォームＳＥＰ１におけるサービス提供源１１０がサービス提供源２１０に変更され、端末１３０が端末２３０に変更されている。

サービス提供源２１０は、サービス提供源１１０におけるサービス識別設定器１１２がメタサービス指定器２１２に交換されている。端末２３０は端末１３０におけるエクセキュータ１３３がエクセキュータ２３３に交換されると共に、リソースセレクタ１３４とサービス依存リソース管理器１３６との間にメタサービス判定器２３４が新たに設けられている。

端末２３０は、第１の実施の形態における端末１３０のダウンローダ１３１がダウンローダ２３１に置き換えられ、識別化実行形式格納器１３２が識別化実行形式格納器２

３２に置き換えられ、エクセキュータ１３３がエクセキュータ２３３に置き換えられると共に、リソースセレクタ１３４とサービス依存リソース管理器１３６との間にメタサービス判定器２３４が新たに設けられている。

サービス提供源２１０において、メタサービス指定器２１２は、上述のサービス識別設定器１１２にメタサービス指定情報生成機能が付与されている。つまり、メタサービス指定器２１２は、サービス識別設定器１１２と同様に、実行形式ＤＥの属性等の副次的な事項を表す付随情報ＩＳを生成すると共に、メタサービスに対応付ける実行形式ＤＥを指定するメタサービス指定情報ＩＳｍを生成する。

メタサービスとは、上述のサービスの一種であるが、実際にユーザの利用するサービスの利用開始／終了などのサービスの機能／諸元の変更や新規サービスの追加といったサービスの拡張を行うことができる唯一のサービスであり、サービスの拡張を目的として存在するものである。この意味において、本明細書においては上述の第１の実施の形態にかかるサービスＳｏと、本実施の形態におけるメタサービスＳｍｅｔａを区別して説明する。

付随情報ＩＳが実行形式ＤＥに付与されて識別化実行形式ＤＥｉが生成され、メタサービス指定情報ＩＳｍが実行形式ＤＥに付与されてメタサービス実行形式ＤＥｍが生成される。つまり、メタサービス指定器２１２からは、識別化実行形式ＤＥｉとメタサービス実行形式ＤＥｍが混在して出力される。そして、これらの識別化実行形式ＤＥｉとメタサービス実行形式ＤＥｍは、送出器１１３およびデリ

バリシステム１２０を介して端末２３０に入力される。そして、端末２３０のダウンローダ２３１によって、識別化実行形式ＤＥｉとメタサービス実行形式ＤＥｍは識別化実行形式格納器２３２に出力される。そして、ダウンローダ２３１は識別化実行形式ＤＥｉから付随情報ＩＳを抽出し、メタサービス実行形式ＤＥｍからメタサービス指定情報ＩＳｍを抽出して、それぞれをサービス依存リソース管理器２３６に出力する。

図９に、メタサービス指定器２１２から出力された識別化実行形式ＤＥｉとメタサービス実行形式ＤＥｍが識別化実行形式格納器２３２に格納される様子を示す。同図に示す例においては、図２に示した第１の実施の形態におけるのと同様に、サービスＳ１、サービスＳ２乃至サービスＳｎ（ｎは任意の自然数）に対応する識別化実行形式ＤＥｉ（Ｓ１）、ＤＥｉ（Ｓ２）、およびＤＥｉ（Ｓｎ）が例示されている。ただし、本図においては、サービスＳ３に対応する識別化実行形式ＤＥｉ（Ｓ３）の位置に、初めてのメタサービス実行形式ＤＥｍ（１）が表示されている。メタサービス実行形式ＤＥｍ（１）は、実行形式ＤＥに、それがメタサービスであることを示すメタサービス指定情報ＩＳｍ（１）が付与されている。

メタサービス実行形式ＤＥｍ（１）は、それがメタサービスｍ１に対応するサービス識別情報Ｅｓ（Ｓｍ１）と副次情報αｍ１から成るメタサービス指定情報ＩＳｍと実行形式ＤＥを含む。具体的には、サービス識別情報Ｅｓ（Ｓｍ１）が実行形式ＤＥとメタサービス１とを対応付けてい

る。このようにサービス識別情報Ｅｓ（Ｓｍ１）の「Ｓｍ１」がメタサービス１を規定する点を除けば、メタサービス指定情報ＩＳｍも基本的には付随情報ＩＳと同じものである。

つまり、メタサービスＳｍは、上述のように、サービスＳｎの１つであるので、メタサービス実行形式ＤＥｍは、識別化実行形式ＤＥｉ（Ｓｍ）と表すこともできる。よって、メタサービス指定情報ＩＳｍも付随情報ＩＳと総称してもよいが、本明細書においては、本実施の形態における特徴をわかりやすくするために、メタサービス指定情報ＩＳｍを付随情報ＩＳと区別して説明する。さらに、メタサービスに対応するサービス識別情報Ｅｓを、メタサービス識別情報Ｅｓｍと称して、実行形式ＤＥに対応するサービス識別情報Ｅｓと区別して説明する。

識別化実行形式格納器２３２の動作は、上述の第１の実施の形態にかかる識別化実行形式格納器１３２と基本的に同じである。ただし、識別化実行形式格納器２３２は、識別化実行形式ＤＥｉとメタサービス実行形式ＤＥｍとを格納し、要求に応じて、識別化実行形式ＤＥｉあるいはメタサービス実行形式ＤＥｍをエクセキュータ２３３に出力する。

エクセキュータ２３３は、識別化実行形式ＤＥｉからサービス識別情報Ｅｓを抽出し、メタサービス実行形式ＤＥｍからサービス識別情報Ｅｓｍを抽出して、それぞれをメタサービス判定器２３４に出力する。

メタサービス判定器２３４は、リソースセレクタ１３４

から入力されるＡＰＩ呼出Ｃａｐｉと、エクセキュータ１３３から入力されるサービス識別情報Ｅｓおよびサービス識別情報Ｅｓｍに基づいて、実行中の実行形式ＤＥがメタサービスに対応付けられている場合にのみ、ＡＰＩ呼出Ｃａｐｉをサービス依存リソース管理器１３６に出力する。

サービス依存リソース管理器１３６は、サービス依存リソースＲＳを格納すると共にサービス依存リソース管理テーブルＴｒｓによってサービス依存リソースＲＳを管理する。なお、サービス依存リソース管理器１３６に格納されるサービス依存リソースＲＳおよびそのサービス依存リソース管理テーブルＴｒｓは、図３を参照して説明した第１の実施の形態におけるものと同じでよい。

そして、サービス依存リソース管理器１３６は、メタサービス判定器２３４を経由して入力されるＡＰＩ呼出Ｃａｐｉに応答して、サービス依存リソース管理器１３６に格納されているサービス依存リソースＲＳへの参照や操作などを行う。また、ダウンローダ１３１から供給される付随情報ＩＳに基づいて、実行形式ＤＥのそれぞれが対応するサービスの内容を認識する。

新たなサービスの追加は、サービス依存リソース管理器１３６が、ダウンローダ２３１から入力される付随情報ＩＳおよびメタサービス指定情報ＩＳｍに基づいて、サービス依存リソース管理テーブルＴｒｓに登録されていない新規のサービスの存在を検出した時点で、そのサービスに対する識別行を追加して、内容を記述する。つまり、サービス依存リソース管理テーブルＴｒｓには、列Ｃ１にはメタ

サービスの内容が書き込まれるが、サービス識別行追加の手順だけに注目すれば、第１の実施の形態における処理内容と同様である。

次に、図１０に示すフローチャートを参照して、端末２３０によるサービス実行ルーチンの動作について説明する。具体的には、端末２３０において、エクセキュータ２３３が呼び出すＡＰＩに対する実行形式ＤＥが実行されることによって、サービス実行が実現される。つまり、エクセキュータ１３３が識別化実行形式格納器１３２から入力される識別化実行形式ＤＥｉおよびメタサービス実行形式ＤＥｍに含まれる実行形式ＤＥを実行させるために、ＡＰＩ呼出Ｃａｐｉを発行した時点で、本ルーチンにおける処理が開始される。

なお、図１０に示すフローチャートは、上述の図５に示すフローチャートにおいて、「実行中の実行形式ＤＥのサービス識別情報Ｅｓを実行エンジンから取得」ステップＳ５１４が「実行中の実行形式ＤＥのサービス識別情報Ｅｓまたはサービス識別情報Ｅｓｍを取得」ステップＳ１０１４に交換され、「実行中のサービスに対する操作かを判定」ステップＳ５１６が「実行中のメタサービスに対する操作かを判定」ステップＳ１０１６に交換されている点を除いては同様に構成されている。以下、本実施の形態に固有の動作に重点をおいて説明する。

よって、ステップＳ５１２において、リソースセレクタ１３４によって、エクセキュータ１３３から発行されたＡＰＩ呼出Ｃａｐｉに基づいて、呼び出されたＡＰＩがサー

ビス依存ＡＰＩであると判断されて、処理はステップＳ１０１４に進む。

ステップＳ１０１４において、メタサービス判定器２３４によって、エクセキュータ２３３から現在実行中の実行形式ＤＥに対するサービス識別情報Ｅｓあるいはメタサービス識別情報Ｅｓｍが読み出される。そして、処理は次のステップＳ１００６に進む。

ステップＳ１０１６において、実行中の実行形式ＤＥがメタサービスに対応付けられているか否かが判断される。Ｎｏの場合、上述のエラー処理ステップＳ５２１を経て、本ルーチンが終了される。一方、実行中の実行形式ＤＥがメタサービスに対応付けられている場合は、Ｙｅｓと判断されて、処理は上述の「サービス依存リソースに対して処理」ステップＳ５１８を経て本ルーチンが終了される。

なお、ステップＳ５１８においては、実行中のメタサービスの実行形式ＤＥであれば（ステップＳ１０１６でＹｅｓ）、全てのサービスに対するサービス依存リソースに対する処理が実行される。

上述のように、第２の実施の形態においては、実行形式ＤＥがメタサービスであれば、全てのサービスに対するサービス依存ＡＰＩが実行できる。従ってメタサービスの実行形式ＤＥの画面上で、サービス提供部から提供可能なサービス一覧を表示し、一覧表示上で、サービスの機能／諸元の変更や新規サービスの追加に代表されるサービスの拡張が達成される。

（第３の実施の形態）

次に、図１１、図１２、および図１３を参照して、本発明の第３の実施の形態にかかるサービス安全拡張プラットフォームについて詳細に説明する。図１１に示すように、本実施の形態にかかるサービス安全拡張プラットフォームＳＥＰ３は、図１に示すサービス安全拡張プラットフォームＳＥＰ１におけるサービス提供源１１０がサービス提供源３１０に変更され、端末１３０が端末３３０に変更されている。

サービス提供源３１０は、サービス提供源１１０におけるサービス識別設定器１１２がサービス識別設定器３１２に交換されると共に、サービス識別親子管理器３１４が新たに設けられている。サービス識別親子管理器３１４は、個々のサービスに親子関係がある場合に、そのような関係を識別情報Ｅｓの間の親子関係として管理する。さらに、管理している親子関係を示すサービス親子指定情報ＩＳｈをサービス識別設定器３１２に出力する。

端末３３０は、端末１３０におけるダウンローダ１３１がダウンローダ３３１に交換され、識別化実行形式格納器１３２が識別化実行形式格納器３３２に交換され、エクセキュータ１３３がエクセキュータ２３３に交換され、親子判定器３３４がリソースセレクタ１３４とサービス依存リソース管理器１３６の間に新たに設けられている。

サービス提供源３１０において、サービス識別親子管理器３１４が管理するサービスの親子関係とは、親サービスが子サービスのサービス依存リソースＲｓを操作可能とする関係と定義される。なお、必要に応じて、親サービスＳ

ｐ、子サービスＳｃ、親サービスのサービス依存リソースＲＳｃを称して、互いに識別するのもとする。例えば、「音楽コンテンツ配信サービス」といったサービスのカテゴリに対応する親サービスに対して、「松下ミュージック配信サービス」や、「テイチクミュージック配信サービス」、および「イーピーチャンネル音楽サービス」に代表される個々のサービスの種類に対応する子サービスがある。

サービス識別設定器３１２は、上述のサービス識別設定器１１２に親子サービス識別情報生成機能が付与されている。つまり、サービス識別設定器３１２は、サービス識別設定器１１２と同様に、実行形式ＤＥの属性等の副次的な事項を表す付随情報ＩＳを生成する。さらに、サービス識別設定器３１２は、サービス識別親子管理器３１４から供給されるサービス親子情報Ｉｈに基づいて、サービス識別設定器３１２は、サービス親子情報Ｅｓｃを生成する。例えば、サービス親子情報Ｉｈに基づいて、サービス親子指定情報ＩＳｈ（Ｓ１－１）が生成される。サービス親子情報Ｅｓｃ（Ｓ１－１）は、このサービスが親サービスＳ１の子サービスであることを定義する。

付随情報ＩＳが実行形式ＤＥに付与されて識別化実行形式ＤＥｉが生成され、サービス親子指定情報ＩＳｈが実行形式ＤＥに付与されて親子識別化実行形式ＤＥｃが生成される。つまり、サービス識別設定器３１２からは、識別化実行形式ＤＥｉと親子識別化実行形式ＤＥｃが混在して出力される。そして、これらの識別化実行形式ＤＥｉと親子識別化実行形式ＤＥｃは、送出器１１３およびデリバリシ

ステム１２０を介して端末３３０に入力される。そして、端末３３０のダウンローダ３３１によって、識別化実行形式ＤＥｉと親子識別化実行形式ＤＥｃは識別化実行形式格納器３３２に出力される。さらに、ダウンローダ３３１は、識別化実行形式ＤＥｉから付随情報ＩＳを抽出し、親子識別化実行形式ＤＥｃからサービス親子指定情報ＩＳｈを、それぞれをサービス依存リソース管理器１３６に出力する。

図１２に、サービス識別設定器３１２から出力された識別化実行形式ＤＥｉと親子識別化実行形式ＤＥｃが識別化実行形式格納器３３２に格納される様子を示す。同図に示す例においては、図２に示した第１の実施の形態におけるのと同様に、サービスＳ１、サービスＳ２乃至サービスＳｎ（ｎは任意の自然数）に対応する識別化実行形式ＤＥｉ（Ｓ１）、ＤＥｉ（Ｓ２）、およびＤＥｉ（Ｓｎ）が例示されている。ただし、本図においては、サービスＳ３に対応する識別化実行形式ＤＥｉ（Ｓ３）の位置に、初めての親子識別化実行形式ＤＥｃ（１）が表示されている。親子識別化実行形式ＤＥｃ（１）は、実行形式ＤＥに、それが子サービスであることを示すサービス親子指定情報ＩＳｈ（１）が付与されている。

親子識別化実行形式ＤＥｃ（１）は、それが親サービスＳｐ１に対応するサービス識別情報Ｅｓ（Ｓｃ１）と副次情報αｃ１から成るサービス親子指定情報ＩＳｈ（１）と実行形式ＤＥを含む。具体的には、親子サービス識別情報Ｅｓｃが実行形式ＤＥを親サービスＳｐ１に対する子サー

ビスＳｃ１として対応付けている。このようにサービス識別情報Ｅｓ（Ｓｐ１）の「Ｓｐ１」が子サービス１を規定する点を除けば、サービス親子指定情報ＩＳｈも基本的には付随情報ＩＳと同じでものである。

つまり、子サービスＳｃは、上述のように、サービスＳｎの１つであるので、親子識別化実行形式ＤＥｃは、識別化実行形式ＤＥｉ（Ｓｃ）と表すこともできる。よって、サービス親子指定情報ＩＳｈも付随情報ＩＳと総称してもよいが、本明細書においては、本実施の形態における特徴をわかりやすくするために、サービス親子指定情報ＩＳｈを付随情報ＩＳと区別して説明する。さらに、親サービスＳｐに対する子サービスＳｃを規定するサービス識別情報Ｅｓを親子サービス識別情報Ｅｓｃと称して、実行形式ＤＥに対応するサービス識別情報Ｅｓと区別して説明する。

識別化実行形式格納器３３２の動作は、上述の第１の実施の形態にかかる識別化実行形式格納器１３２と基本的に同じである。ただし、識別化実行形式格納器３３２は、識別化実行形式ＤＥｉと親子識別化実行形式ＤＥｃとを格納し、要求に応じて、識別化実行形式ＤＥｉあるいは親子識別化実行形式ＤＥｃをエクセキュータ３３３に出力する。

エクセキュータ３３３は、識別化実行形式ＤＥｉからサービス識別情報Ｅｓを抽出し、親子識別化実行形式ＤＥｃから親子サービス識別情報Ｅｓｃを抽出して、それぞれを親子判定器３３４に出力する。

親子判定器３３４は、リソースセレクタ１３４から入力されるＡＰＩ呼出Ｃａｐｉと、エクセキュータ３３３から

入力されるサービス識別情報Ｅｓおよび親子サービス識別情報Ｅｓｃ基づいて、実行中の実行形式ＤＥがＡＰＩ呼出Ｃａｐｉの処理対象のサービス依存リソースのサービスの先祖（親、または親の親、または親の親の親．．．）である場合にのみ、ＡＰＩ呼出Ｃａｐｉをサービス依存リソース管理器１３６に出力する。

つまり、親子判定器３３４は、実行中の実行形式ＤＥの親子サービス識別情報Ｅｓｃをエクセキュータ３３３から得るとともに、エクセキュータ３３３から発行されるＡＰＩ呼出Ｃａｐｉをリソースセレクタ１３４を介して得る。さらに、ダウンローダ３３１からサービス間の親子関係を判定するための情報であるサービス親子指定情報ＩＳｈを得る。そして、親子サービス識別情報Ｅｓｃが示すサービスが、サービス親子指定情報ＩＳｈの示すサービスの先祖であるかを判断する。そして、先祖であると判断した場合には、サービス依存リソース管理器１３６に対してサービス依存ＡＰＩの呼び出し要求（ＡＰＩ呼出Ｃａｐｉ）を伝え、先祖でないと判断した場合には伝えない。

次に、図１３に示すフローチャートを参照して、端末３３０におけるサービス実行ルーチンの動作について説明する。具体的には、端末３３０において、エクセキュータ３３３が呼び出すＡＰＩに対する実行形式ＤＥおよび親子識別化実行形式ＤＥｃに含まれる実行形式ＤＥを実行させるために、ＡＰＩ呼出Ｃａｐｉを発行した時点で、本ルーチンにおける処理が開始される。

なお、図１３に示すフローチャートは、上述の図５に示

すフローチャートにおいて、「実行中の実行形式ＤＥのサービス識別情報Ｅｓを実行エンジンから取得」ステップＳ５１４が「実行中の実行形式ＤＥのサービス識別情報Ｅｓまたは親子サービス識別情報Ｅｓｃを取得」ステップＳ１３１４に交換され、「実行中のサービスに対する操作かを判定」ステップＳ５１６が「実行中のメタサービスに対する操作かを判定」ステップＳ１３１６に交換されている点を除いては同様に構成されている。以下、本実施の形態に固有の動作に重点をおいて説明する。

よって、ステップＳ５１２において、リソースセレクタ１３４によって、エクセキュータ３３３から発行されたＡＰＩ呼出Ｃａｐｉに基づいて、呼び出されたＡＰＩがサービス依存ＡＰＩであると判断されて、処理はステップＳ１３０４に進む。

ステップＳ１３１４において、親子判定器３３４によって、エクセキュータ３３３から現在実行中の実行形式ＤＥに対するサービス識別情報Ｅｓあるいは親子サービス識別情報Ｅｓｃが読み出される。そして、処理は次のステップＳ１３０６に進む。

ステップＳ１３１６において、親子判定器３３４によって、サービス依存ＡＰＩ呼出Ｃａｐｉの処理対象のサービス依存リソースに対応するサービスに対して、実行中の実行形式ＤＥの対応するサービスが先祖のサービスであるか否かが判断される。Ｎｏの場合、上述のエラー処理ステップＳ５２１を経て、本ルーチンが終了される。一方、実行中の実行形式ＤＥが実行中のサービスが先祖である場合は

、Ｙｅｓと判断されて、処理は上述の「サービス依存リソースに対して処理」ステップＳ５１８を経て本ルーチンが終了される。

なお、ステップＳ５１８においては、サービス依存リソース管理器１３６によって、サービス依存ＡＰＩ呼出Ｃａｐｉに対応するサービス依存リソースに対する処理が実行される。すなわち、現在実行している実行形式ＤＥのサービスの子孫（子、または子の子、または子の子の子、．．．）のサービス依存リソースに対して処理が可能になるように管理される。

（第４の実施の形態）

次に、図１４および図１５、図１６、および図１７を参照して、本発明の第４の実施の形態にかかるサービス安全拡張プラットフォームについて説明する。図８に示すように、本実施の形態の形態にかかるサービス安全拡張プラットフォームＳＥＰ４は、図１に示すサービス安全拡張プラットフォームＳＥＰ１におけるサービス提供源１１０がサービス提供源４１０に変更され、端末１３０が端末４３０に変更されている。

サービス提供源４１０は、サービス提供源１１０における実行形式提供器１１１がコンテンツ提供器４１１に交換され、サービス識別設定器１１２が制御コンテンツ指定器４１２に交換されている。端末４３０は、端末１３０におけるダウンローダ１３１がコンテンツダウンローダ４３１に交換され、識別化実行形式格納器１３２がコンテンツ格納器４３２に交換され、エクセキュータ１３３がエクセキ

ュータ４３３に交換されると共に、リソースセレクタ１３４とサービス依存リソース管理器１３６との間に制御コンテンツ判定器４３４が新た設けられている。

コンテンツ提供器４１１は、サービスを実現する実行形式ＤＥと実行形式ＤＥが解釈しユーザに提示する目的で製作されたデータであるコンテンツＤＣを格納し、必要に応じて出力する。実行形式ＤＥもコンテンツＤＣの満たすべき要件を備えている。例えば、共に、実行形式ＤＥもコンテンツＤＣも共に、逐次動作手順をプログラミングするスクリプト記述を含めることが可能なマークアップ言語で記述する。スクリプトの言語としてＪａｖａ（Ｒ）系のＪａｖａＳｃｒｉｐｔやＥＣＭＡＳｃｒｉｐｔなどの言語でよい。また、マークアップ言語としては、ＨＴＭＬ言語やＢＭＬ言語でよい。

制御コンテンツ指定器４１２は、コンテンツ提供器４１１が出力するコンテンツＤＣのうち、制御コンテンツに対して制御コンテンツであることを指定する付随情報ＩＳＣを付与して制御コンテンツＤＣｃとして出力する。制御コンテンツＤＣｃの定義は、サービスに対応付けられた実行形式ＤＥとして製作されたコンテンツＤＣである。

コンテンツダウンローダ４３１は、デリバリシステム１２０から伝送されてくる、コンテンツＤＣと制御コンテンツＤＣｃとを受信し、コンテンツ格納器４３２に受信した情報を書き込む。

デリバリシステム１２０からは実行形式ＤＥもコンテンツＤＣの一種類として伝送されるため、コンテンツダウン

ローダ４３１は、コンテンツＤＣを格納する機能を有していればよい。

コンテンツダウンローダ４３１は、サービス依存リソース管理器１３６がサービス依存リソースとして保持する、サービスのコンテンツを自動ダウンロードすべきかを表す情報に基づきコンテンツの格納を制御する。

すなわち、コンテンツダウンローダ４３１は、注目するサービスに対し、自動ダウンロードすべき場合には、サービスに関するコンテンツＤＣを制御コンテンツＤＣｃも含め全て格納する。一方、自動ダウンロードすべきでない場合には、制御コンテンツＤＣｃのみを格納する。

コンテンツ格納器４３２は、コンテンツＤＣを格納する。第１の実施の形態の識別化実行形式格納器１３２が格納する情報が実行形式ＤＥであるのに対し、コンテンツ格納器４３２はコンテンツＤＣを格納する。なお、コンテンツＤＣには制御コンテンツＤＣｃが含まれる。

図１５に、制御コンテンツ指定器４１２から出力されたコンテンツＤＣと付随情報ＩＳｃがコンテンツ格納器４３２に格納される様子を示す。同図に示す例においては、サービスＳ１に対して、制御コンテンツＤＣｃであるＣ（Ｓ１，ＤＥ）、コンテンツＤＣであるＣ（Ｓ１，１）、Ｃ（Ｓ１，２）、およびＣ（Ｓ１，３）がコンテンツ格納器４３２に保持されている。制御コンテンツＤＣｃであるＣ（Ｓ１，Ｅ）をエクセキュータ４３３で実行させることで、コンテンツＤＣであるＣ（Ｓ１，１）、Ｃ（Ｓ１，２）、およびＣ（Ｓ１，３）を読み込んでユーザに提示する。す

なわち制御コンテンツＤＣｃであるＣ（Ｓ１，ＤＥ）は、コンテンツＤＣであるＣ（Ｓ１，１）、Ｃ（Ｓ１，２）、Ｃ（Ｓ１，３）に対するいわゆるブラウザと同等の動作を行う。

制御コンテンツ判定器４３４は、エクセキュータ４３３の出力から、実行しているコンテンツＤＣが制御コンテンツＤＣｃであるかを判定する。そして、制御コンテンツＤＣｃである場合のみに、リソースセレクタ１３４からのサービス依存ＡＰＩの要求をサービス依存リソース管理器１３６に伝える。

図１６に、本実施の形態において、サービス依存リソース管理器１３６が格納するサービス依存リソーステーブルＴｒｓ４の一例である。サービス依存リソーステーブルＴｒｓ４は、図３に示した、第１の実施の形態にかかるサービス依存リソース管理テーブルＴｒｓと類似しているが、列Ｃ２には、コンテンツダウンローダ４３１がコンテンツＤＣを自動ダウンロードすべきかについての情報がサービス毎に保持されている。

次に、図１７に示すフローチャートを参照して、端末４３０における制御コンテンツＤＣｃの実行ルーチンの動作について説明する。具体的には、端末４３０において、エクセキュータ４３３が呼び出すＡＰＩに対する制御コンテンツＤＣｃを実行するために、ＡＰＩ呼出Ｃａｐｉを発行した時点で本ルーチンにおける処理が開始される。

なお、図１７に示すフローチャートは、上述の図５に示すフローチャートにおいて、実行中の実行形式ＤＥのサー

ビス識別情報Ｅｓを実行エンジンから取得」ステップＳ５１４が「実行中のコンテンツＤＣのサービス識別情報Ｅｓを取得」ステップＳ１７１４に交換され、「実行中のサービスに対する操作かを判定」ステップＳ５１６が「実行中のサービスに対する制御コンテンツ操作かを判定」ステップＳ１７１６に交換されている点を除いては同様に構成されている。以下、本実施の形態に固有の動作に重点をおいて説明する。

ステップＳ５１２において、リソースセレクタ１３４によって、エクセキュータ３３３から発行されたＡＰＩ呼出Ｃａｐｉに基づいて、呼び出されたＡＰＩがサービス依存ＡＰＩであると判断されて、処理はステップＳ１７０４に進む。

ステップＳ１７１４において、制御コンテンツ判定器４３４によって、エクセキュータ４３３から現在実行中のコンテンツＤＣに対するサービス識別情報Ｅｓが読み出される。そして、処理は次のステップＳ１７１６に進む。

ステップＳ１７１６において、制御コンテンツ判定器４３４によって、サービス依存ＡＰＩ呼出Ｃａｐｉの処理対象のサービス依存リソースに対応するサービスに対して、実行中のコンテンツＤＣがコンテンツＤＣの対応するサービスに対応する制御コンテンツであるか否かが判断される。Ｎｏの場合、上述のエラー処理ステップＳ５２１を経て、本ルーチンが終了される。一方、実行中のコンテンツＤＣが実行中のサービスが先祖である場合は、Ｙｅｓと判断されて、処理は上述の「サービス依存リソースに対して処

理」ステップＳ５１８を経て本ルーチンが終了される。

なお、ステップＳ５１８においては、サービス依存リソース管理器１３６によって、サービス依存ＡＰＩ呼出Ｃａｐｉに対応するサービス依存リソースに対する処理が実行される。

本明細書で開示したサービス安全拡張プラットフォームによれば、サービスを実現する実行形式ＤＥによる不正アクセスの排除を、実行形式ＤＥに不正アクセスを引き起こす処理が含まれることを否定する必要無く達成するという、新次元の安全性を実現することができる。

詳述すれば、他のサービスの状態変更やデータの破壊や、プラットフォーム自体のシステムダウンなど、サービスの正常な享受に悪影響を及ぼす挙動を引き起こす処理である不正アクセスの排除を、実行形式ＤＥに不正アクセスを引き起こす処理が含まれることを実行形式ＤＥの配布元の確からしさから推測するまでもなく達成する。

さらには上記の性質と同時に、実行形式ＤＥの変更や追加によってサービスの機能／諸元の変更や新規サービスの追加といったサービスの拡張も達成する機能を実現することができる。

これにより、従来のサービスの拡張可能なプラットフォームに対して従来にない新次元の安全性と操作性とを同時に実現することができる。

産業上の利用可能性

以上のように、本発明は、「サービスの拡張性」、「サ

ービス安全性の確保」および「サービス拡張操作のシームレス」という3つの特徴を実現する従来にない全く新しいサービス安全拡張プラットフォームに有用である。

請求の範囲

1．サービス（Ｓ）と実行形式（ＤＥ）とが対応付けられており、前記実行形式（ＤＥ）の変更や追加によって前記サービス（Ｓ）の拡張が達成されるサービス安全拡張プラットフォーム（ＳＥＰ）であって、前記サービスの拡張を行うサービス依存ＡＰＩを具備し、かつ前記実行形式（ＤＥ）からの前記サービスの拡張は前記サービス依存ＡＰＩの呼び出しによってのみ行われることを特徴とする、サービス安全拡張プラットフォーム（ＳＥＰ）。

2．前記サービスの拡張が新規サービスの新設であることを特徴とする、請求項１に記載のサービス安全拡張プラットフォーム（ＳＥＰ１）。

3．前記サービスの拡張がサービス利用開始であることを特徴とする、請求項１に記載のサービス安全拡張プラットフォーム（ＳＥＰ１）。

4．注目する前記サービスの拡張は、注目する前記サービスに対応付けられた実行形式（ＤＥ）からの前記サービス依存ＡＰＩの呼び出しによってのみ行われることを特徴とする、請求項１乃至請求項３に記載のサービス安全拡張プラットフォーム（ＳＥＰ）。

5．複数の前記サービス間に親子関係（サービス親子指定情報ＩＳｈ）が定義され、前記実行形式（ＤＥ）が要求する前記サービス依存ＡＰＩ呼び出しが前記サービス（Ｓ）に対応付けられるサービス依存リソース（ＲＳ）を処理対象として指定した際に、前記実行形式（ＤＥ）に対応付け

られるサービス（Ｓ）が前記サービス（Ｓ）の先祖である場合にのみ、サービス依存リソース（ＲＳ）に対して処理可能であることを特徴とする、請求項１乃至請求項３に記載のサービス安全拡張プラットフォーム（ＳＥＰ３）。

６．メタサービスに対応付けられた前記実行形式（ＤＥ）が前記サービス依存ＡＰＩによって、前記サービス（Ｓ）の少なくとも１つが拡張可能なことを特徴とする、請求項１乃至請求項３に記載のサービス安全拡張プラットフォーム（ＳＥＰ）。

７．前記メタサービスに対応付けられた前記実行形式が前記サービス依存ＡＰＩによって、前記サービス（Ｓ）の全てが拡張可能であり、前記メタサービスに対応付けられていない前記実行形式（ＤＥ）は前記サービス依存ＡＰＩによって前記サービスの拡張が不可能であることを特徴とする、請求項６に記載のサービス安全拡張プラットフォーム。

８．前記実行形式（ＤＥ）がコンテンツ（ＤＣ）として満たすべき条件を満たした制御コンテンツ（ＤＣｃ）であり、前記制御コンテンツ（ＤＣ）が、前記コンテンツの少なくとも１つと共にコンテンツ（ＤＣ）として伝送され、前記コンテンツ（ＤＣ）の少なくとも１つから前記制御コンテンツ（Ｄ Ｃ ｃ）を指定する情報（Ｅｓ）が伝送され、前記制御コンテンツ（ＤＣ）によってのみサービス依存ＡＰＩの処理が可能であることを特徴とする請求項１乃至請求項３に記載のサービス安全拡張プラットフォーム（ＳＥＰ４）。

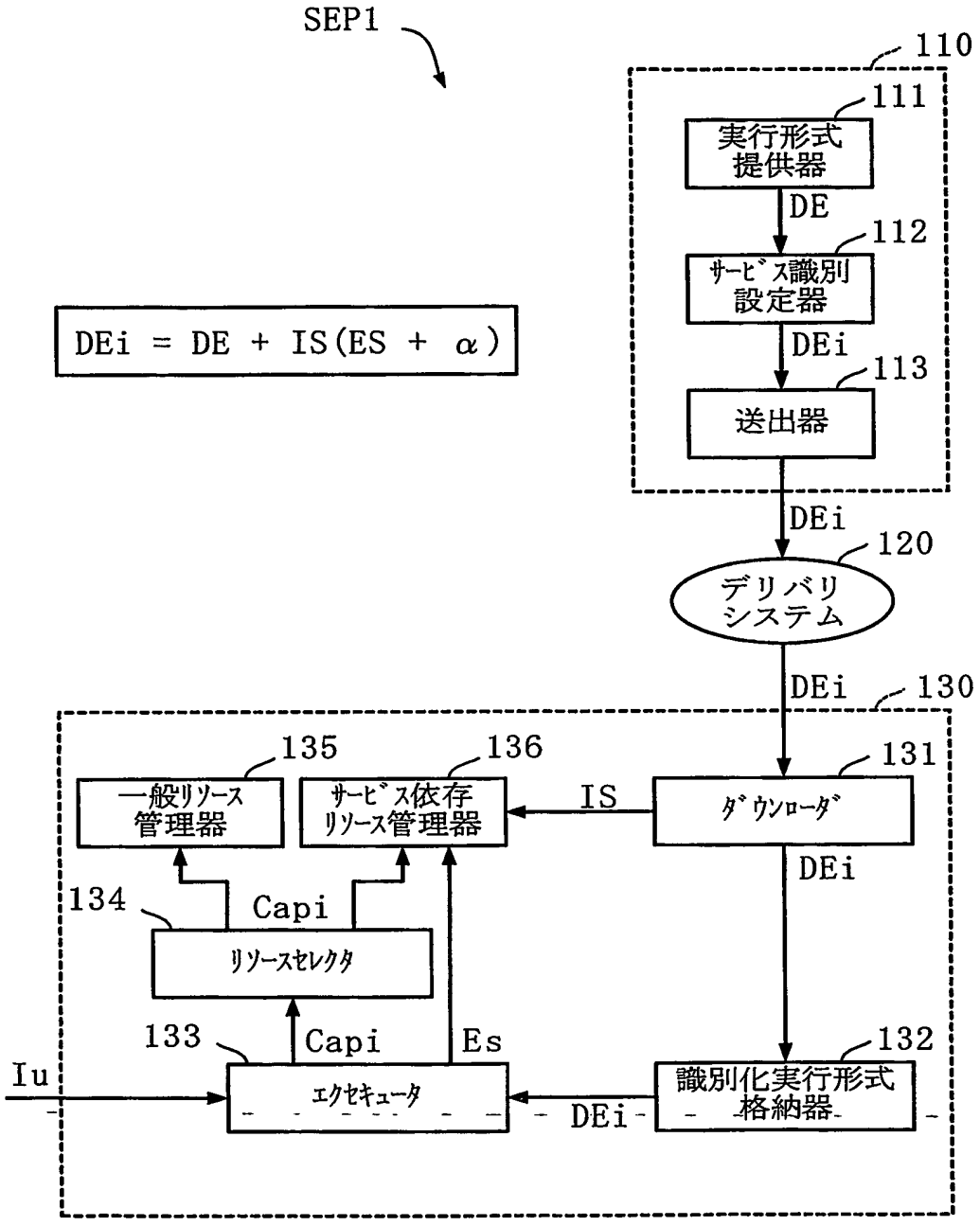
９．前記サービス依存ＡＰＩによって、特定の前記サービスのコンテンツ（ＤＣ）の自動的な格納が制御されることを特徴とする請求項８に記載のサービス安全拡張プラットフォーム（ＳＥＰ４）。

１０．前記実行形式（ＤＥ）を少なくとも１つのサービス提供部（１１０、２１０、３１０、４１０）から送出し、前記実行形式（ＤＥ）を実行する少なくとも１つの端末（１３０、２３０、３３０、４３０）で受信することを特徴とする、請求項１乃至請求項９に記載のサービス安全拡張プラットフォーム（ＳＥＰ）。

１１．請求項１乃至請求項１０に記載のサービス安全拡張プラットフォーム（ＳＥＰ）を実現するサービス安全拡張方法。

１２．請求項１乃至請求項１０に記載のサービス安全拡張プラットフォーム実施するコンピュータプログラムを格納した記憶媒体。

図 1



差替え用紙 (規則26)

図 2

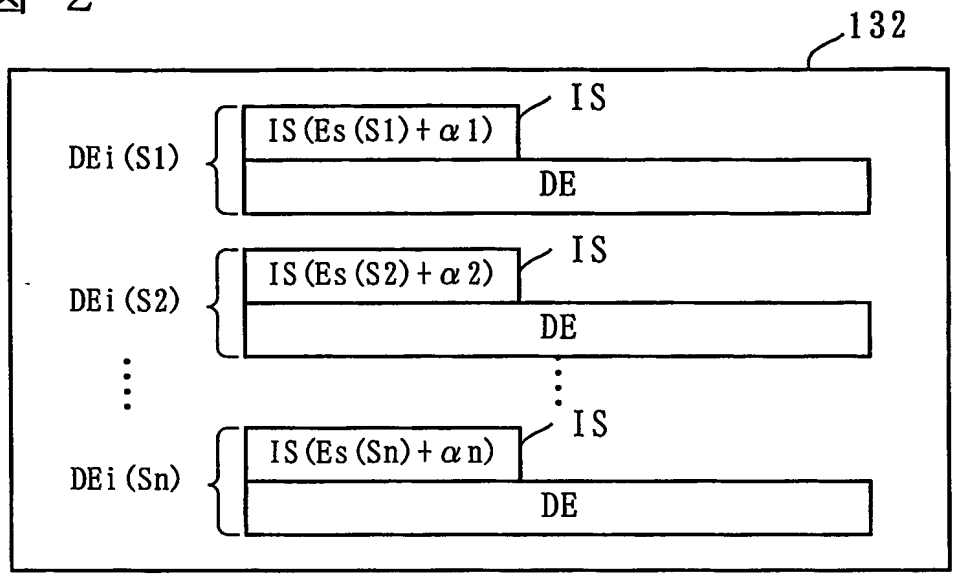

132
IS
IS(Es(S1)+α1)
DE
DEi(S1)
IS
IS(Es(S2)+α2)
DE
DEi(S2)
IS
IS(Es(Sn)+αn)
DE
DEi(Sn)

図 3

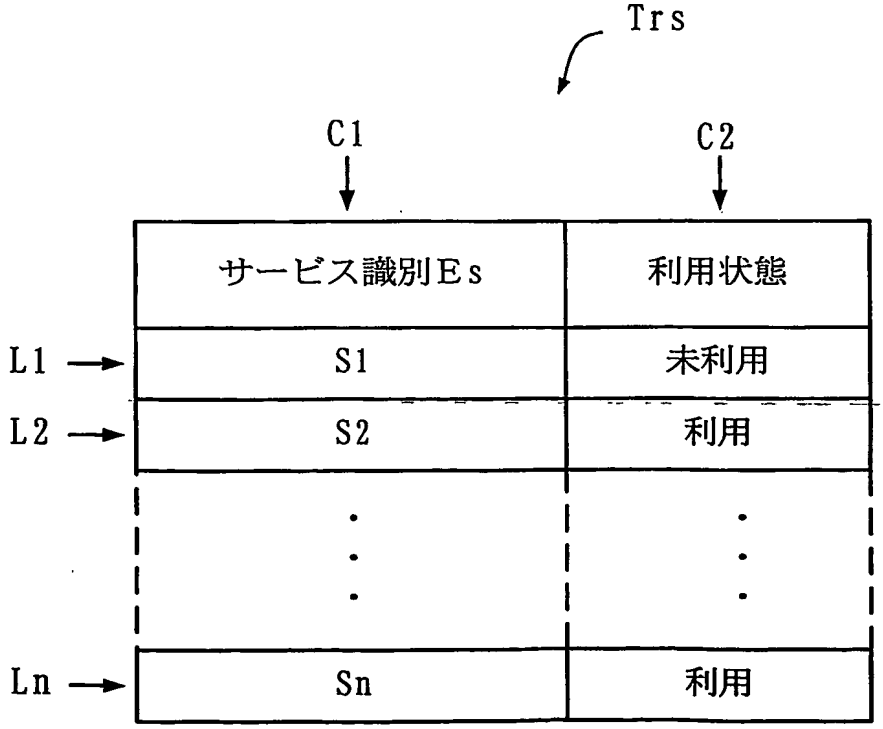

Trs
C1
C2
サービス識別Es
利用状態
L1
S1
未利用
L2
S2
利用
Ln
Sn
利用

図 4

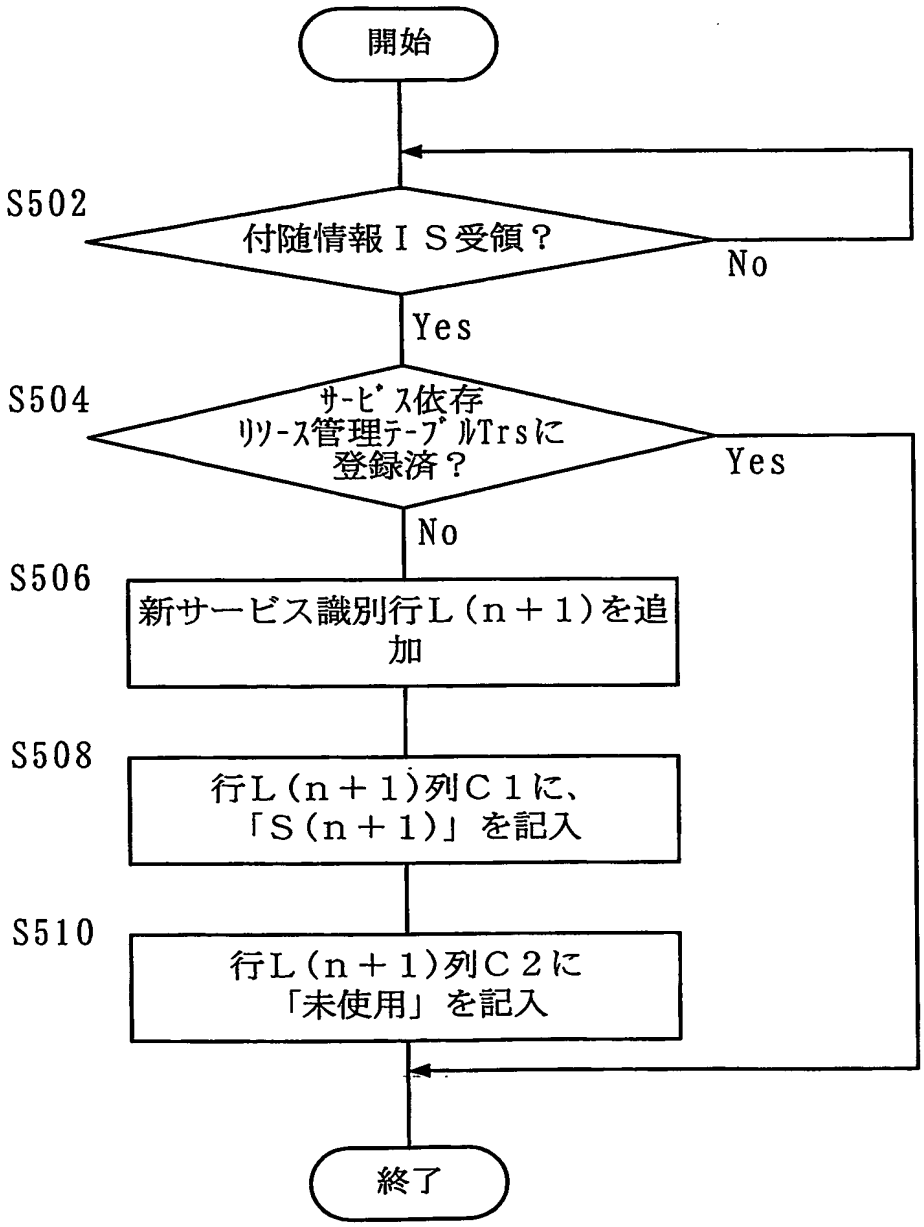

開始
S502
付随情報ＩＳ受領？
No
Yes
S504
ｻｰﾋﾞｽ依存
ﾘｿｰｽ管理ﾃｰﾌﾞﾙTrsに
登録済？
Yes
No
S506
新サービス識別行Ｌ（ｎ＋１）を追加
S508
行Ｌ（ｎ＋１）列Ｃ１に、
「Ｓ（ｎ＋１）」を記入
S510
行Ｌ（ｎ＋１）列Ｃ２に
「未使用」を記入
終了

図 5

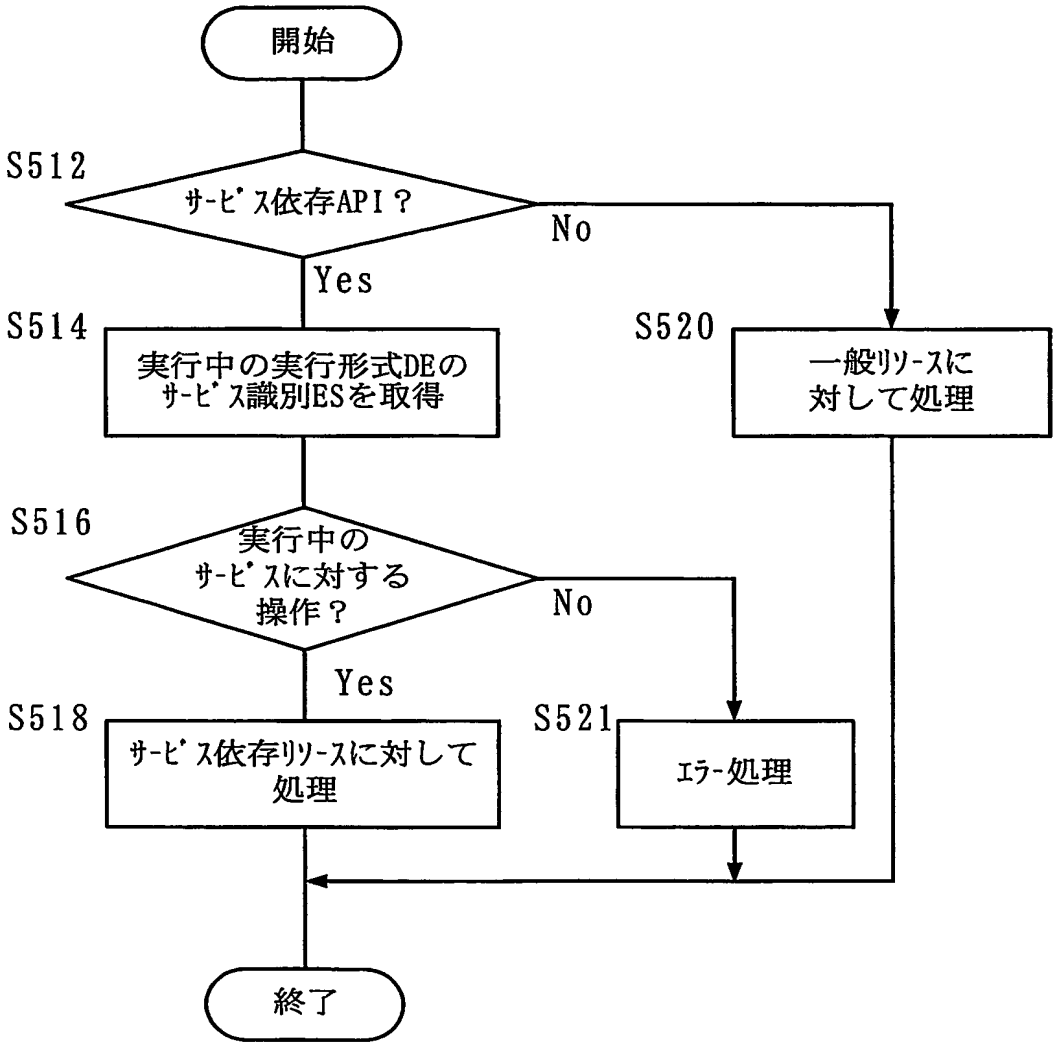

開始
S512
ｻｰﾋﾞｽ依存API？
No
Yes
S514
実行中の実行形式DEの
ｻｰﾋﾞｽ識別ESを取得
S520
一般ﾘｿｰｽに
対して処理
S516
実行中の
ｻｰﾋﾞｽに対する
操作？
No
Yes
S518
ｻｰﾋﾞｽ依存ﾘｿｰｽに対して
処理
S521
ｴﾗｰ処理
終了

図 6

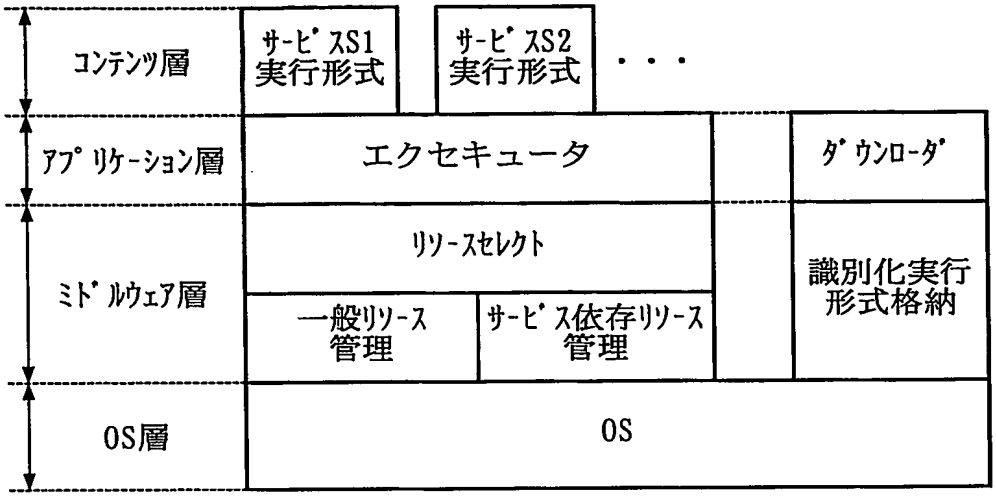

ｺﾝﾃﾝﾂ層
ｻｰﾋﾞｽS1 実行形式
ｻｰﾋﾞｽS2 実行形式
・・・
ｱﾌﾟﾘｹｰｼｮﾝ層
エクセキュータ
ﾀﾞｳﾝﾛｰﾀﾞ
ﾐﾄﾞﾙｳｪｱ層
ﾘｿｰｽｾﾚｸﾄ
一般ﾘｿｰｽ 管理
ｻｰﾋﾞｽ依存ﾘｿｰｽ 管理
識別化実行 形式格納
OS層
OS

図 7

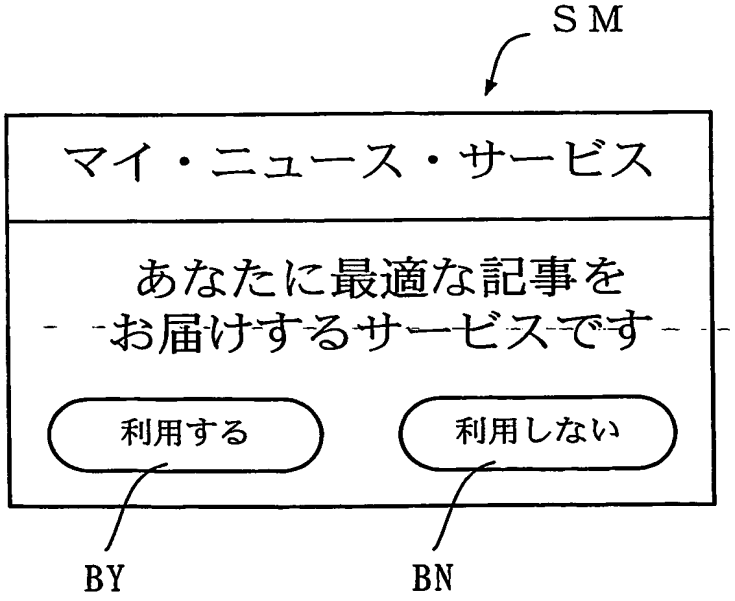

SM
マイ・ニュース・サービス
あなたに最適な記事を
お届けするサービスです
利用する
利用しない
BY
BN

図 8

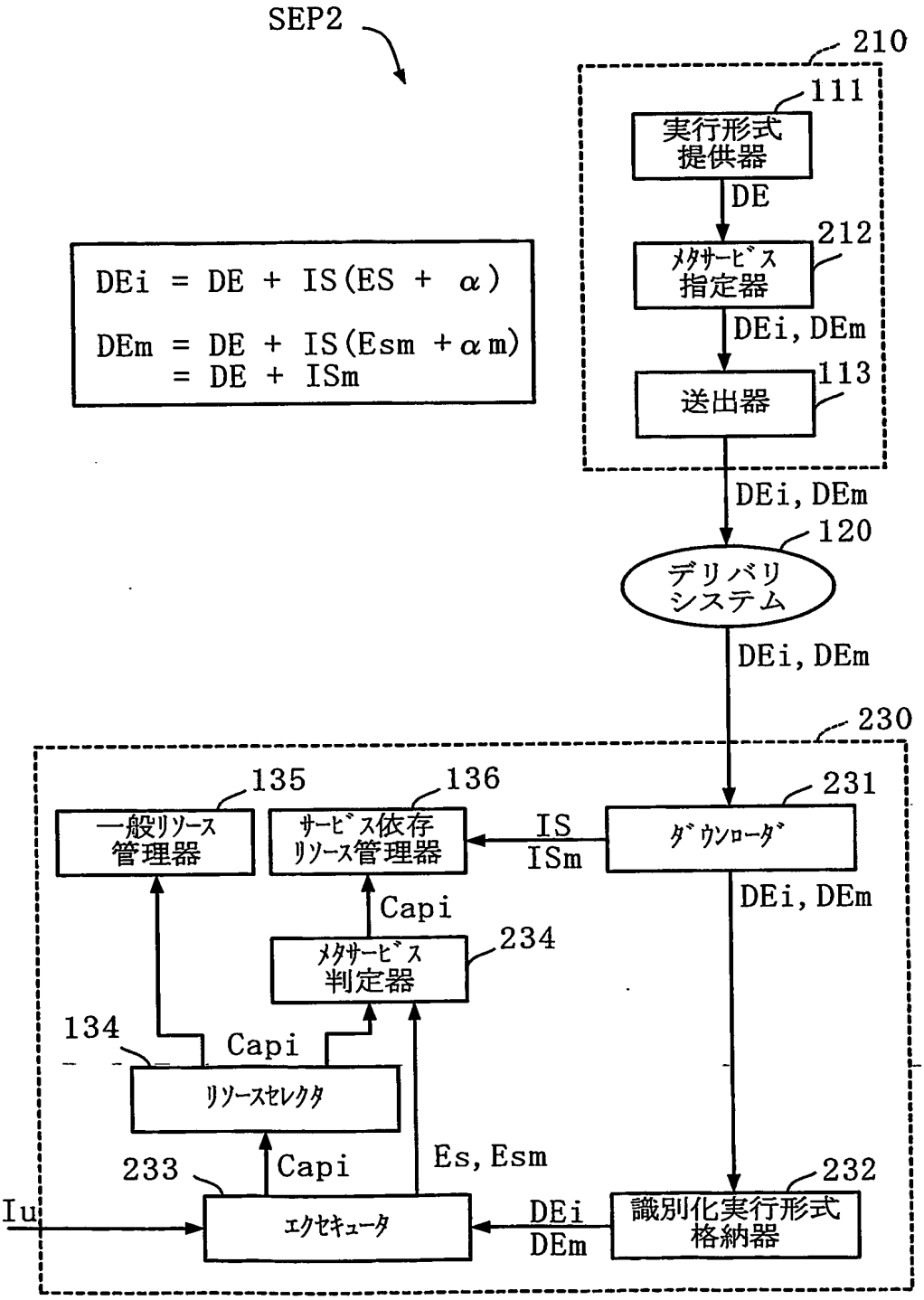

SEP2
DEi = DE + IS(ES + α)
DEm = DE + IS(Esm + αm)
= DE + ISm
210
111
実行形式提供器
DE
212
ﾒﾀｻｰﾋﾞｽ指定器
DEi, DEm
113
送出器
DEi, DEm
120
デリバリシステム
DEi, DEm
230
135
一般ﾘｿｰｽ管理器
136
ｻｰﾋﾞｽ依存ﾘｿｰｽ管理器
IS
ISm
231
ﾀﾞｳﾝﾛｰﾀﾞ
DEi, DEm
Capi
234
ﾒﾀｻｰﾋﾞｽ判定器
134
Capi
ﾘｿｰｽｾﾚｸﾀ
233
Capi
Es, Esm
Iu
ｴｸｾｷｭｰﾀ
DEi
DEm
232
識別化実行形式格納器

図 9

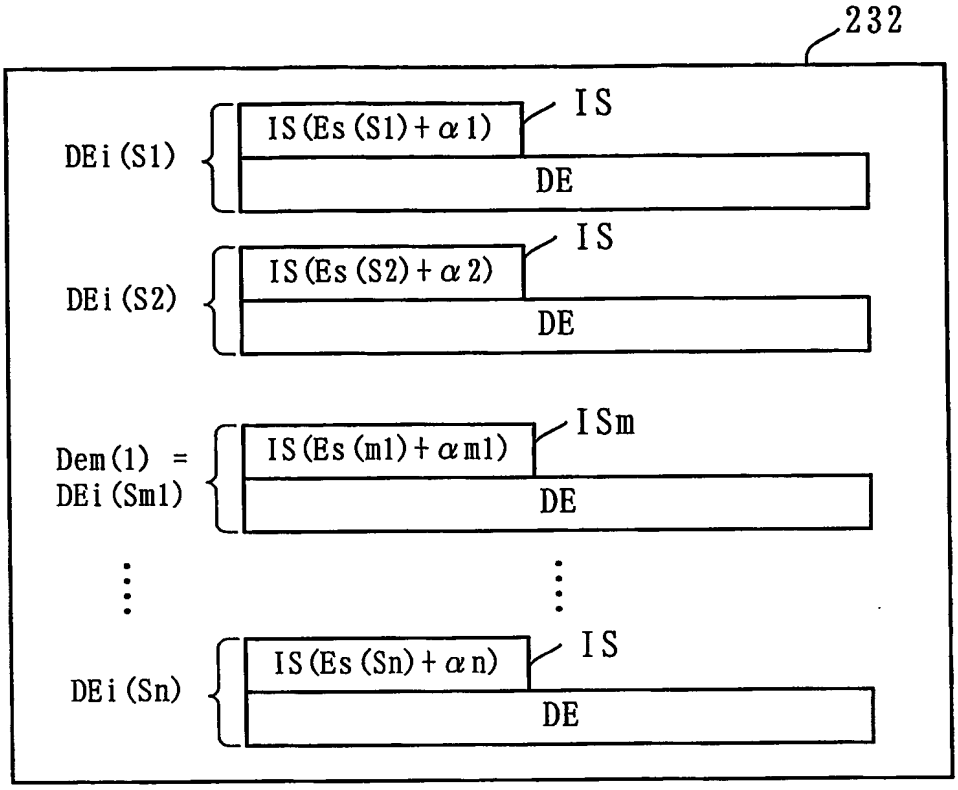

232
DEi(S1)
IS(Es(S1)+α1)
IS
DE
DEi(S2)
IS(Es(S2)+α2)
IS
DE
Dem(1) =
DEi(Sm1)
IS(Es(m1)+αm1)
ISm
DE
DEi(Sn)
IS(Es(Sn)+αn)
IS
DE

図 １０

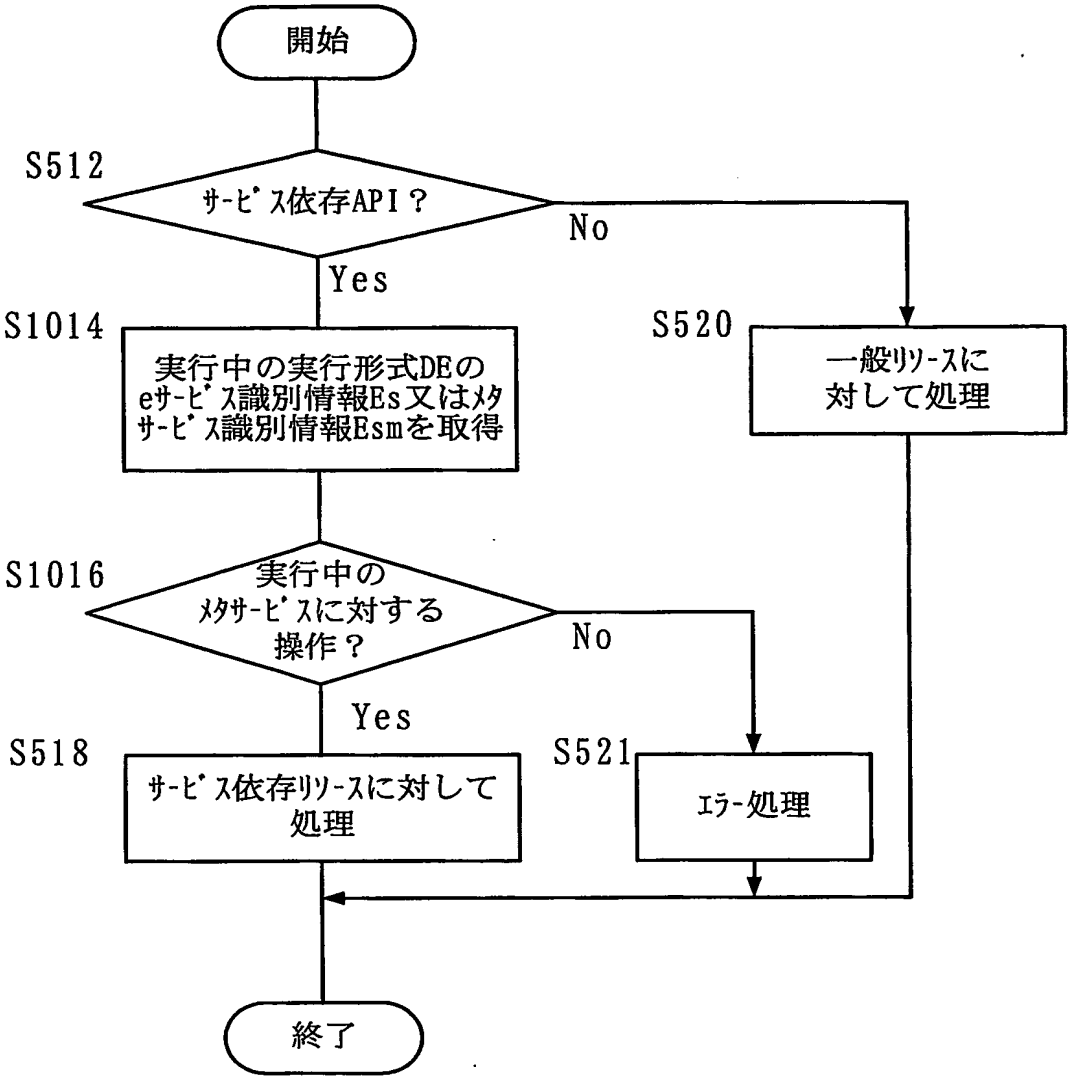

開始
S512
ｻｰﾋﾞｽ依存API？
No
Yes
S1014
実行中の実行形式DEの
eｻｰﾋﾞｽ識別情報Es又はﾒﾀ
ｻｰﾋﾞｽ識別情報Esmを取得
S520
一般ﾘｿｰｽに
対して処理
S1016
実行中の
ﾒﾀｻｰﾋﾞｽに対する
操作？
No
Yes
S518
ｻｰﾋﾞｽ依存ﾘｿｰｽに対して
処理
S521
ｴﾗｰ処理
終了

図 11

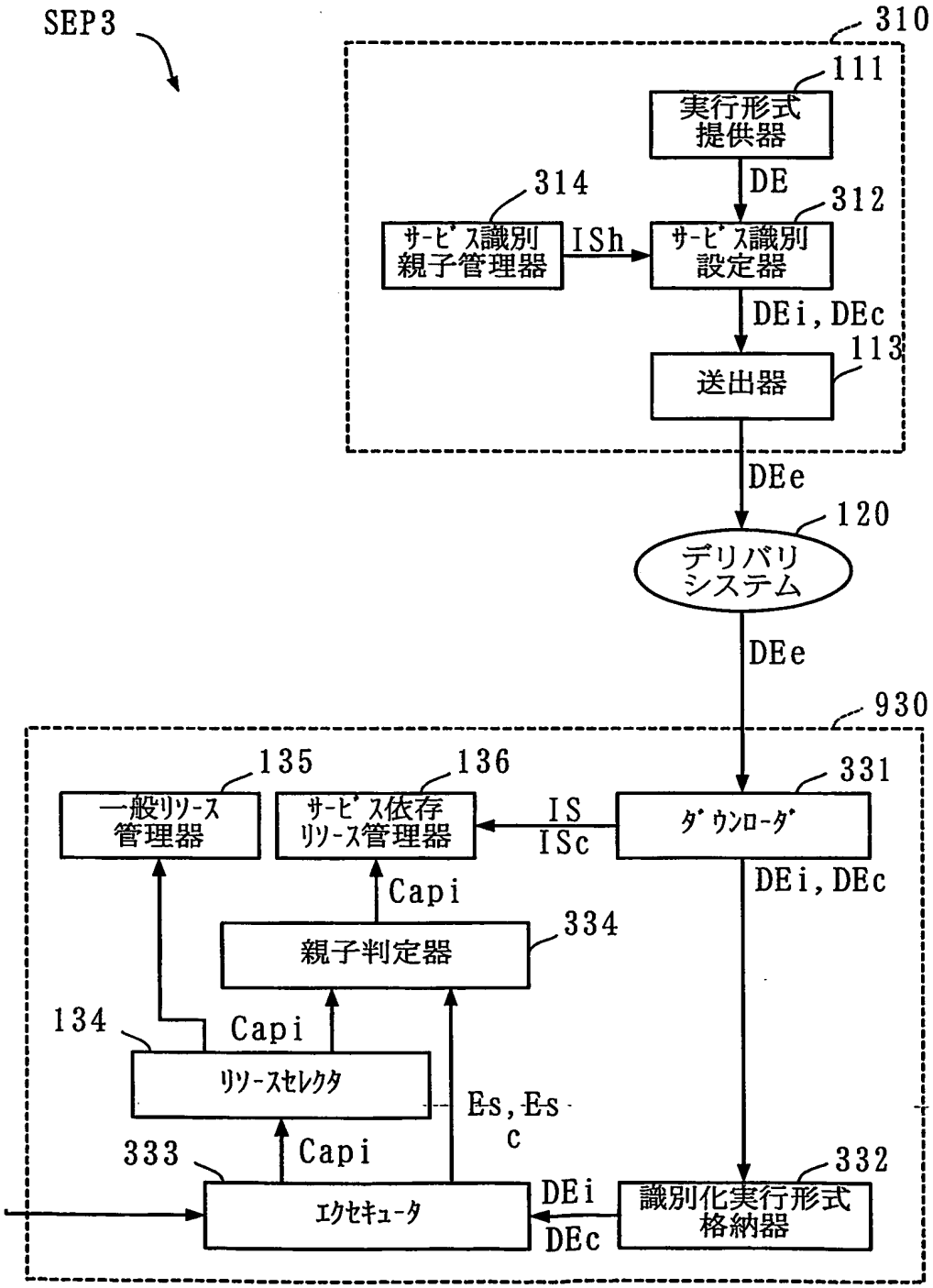

SEP3
310
111
実行形式
提供器
DE
314
312
ｻｰﾋﾞｽ識別
親子管理器
ISh
ｻｰﾋﾞｽ識別
設定器
DEi, DEc
113
送出器
DEe
120
デリバリ
システム
DEe
930
135
136
331
一般ﾘｿｰｽ
管理器
ｻｰﾋﾞｽ依存
ﾘｿｰｽ管理器
IS
ISc
ﾀﾞｳﾝﾛｰﾀﾞ
DEi, DEc
Capi
334
親子判定器
134
Capi
ﾘｿｰｽｾﾚｸﾀ
Es, Esc
333
Capi
332
DEi
DEc
ｴｸｾｷｭｰﾀ
識別化実行形式
格納器

図１２

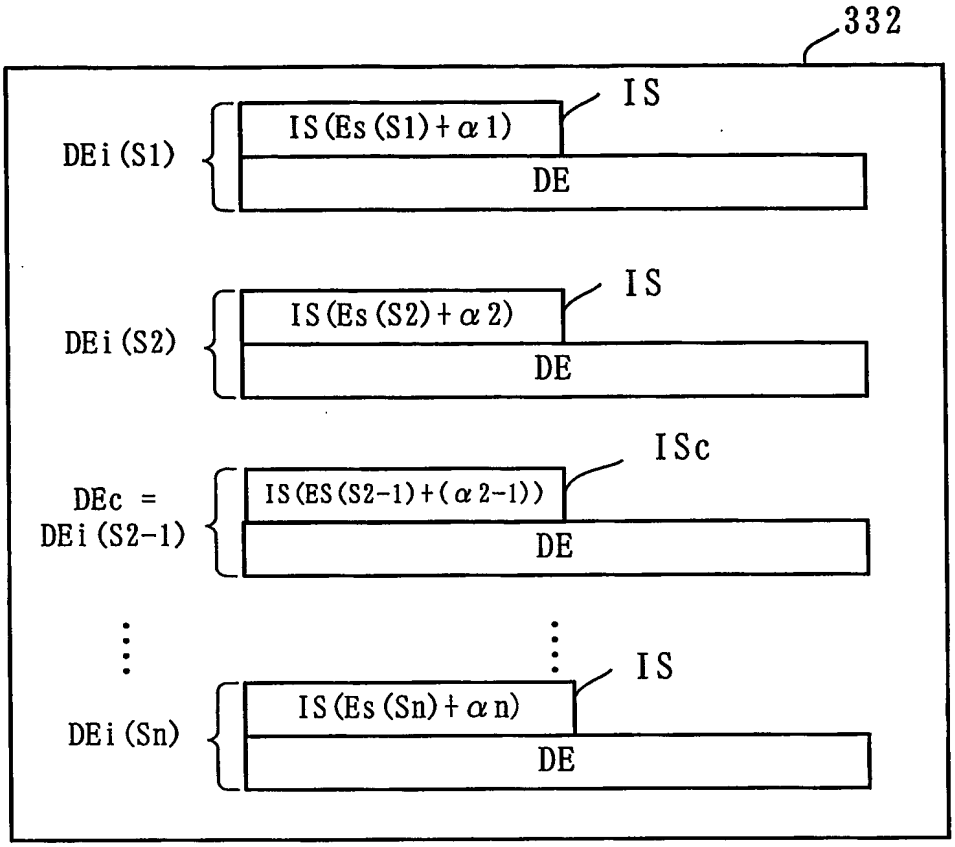

332
IS
IS(Es(S1)+α1)
DE
DEi(S1)
IS
IS(Es(S2)+α2)
DE
DEi(S2)
ISc
IS(ES(S2-1)+(α2-1))
DE
DEc =
DEi(S2-1)
IS
IS(Es(Sn)+αn)
DE
DEi(Sn)

図 １３

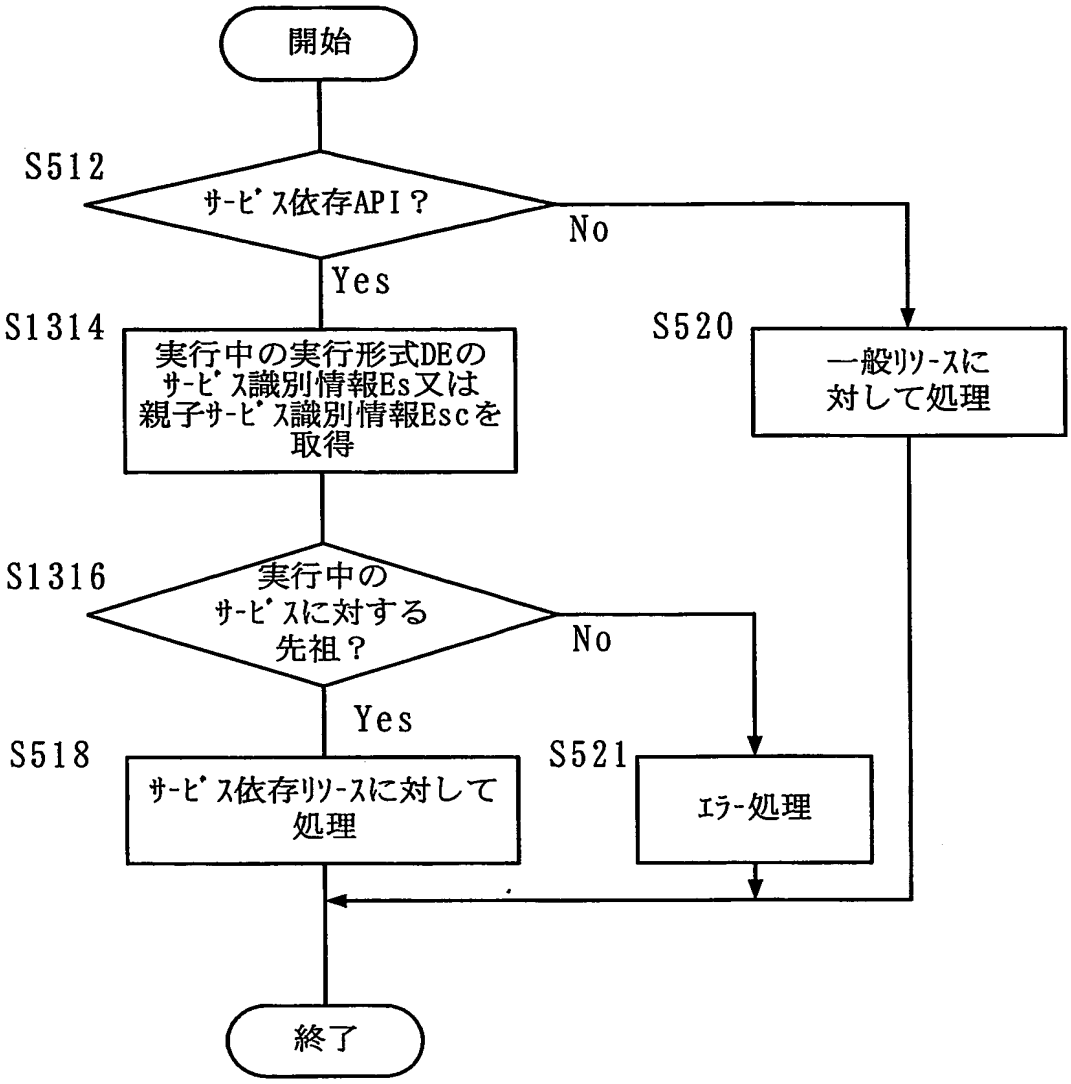

開始
S512
ｻｰﾋﾞｽ依存API？
No
Yes
S1314
実行中の実行形式DEの
ｻｰﾋﾞｽ識別情報Es又は
親子ｻｰﾋﾞｽ識別情報Escを
取得
S520
一般ﾘｿｰｽに
対して処理
S1316
実行中の
ｻｰﾋﾞｽに対する
先祖？
No
Yes
S518
ｻｰﾋﾞｽ依存ﾘｿｰｽに対して
処理
S521
ｴﾗｰ処理
終了

図 １４

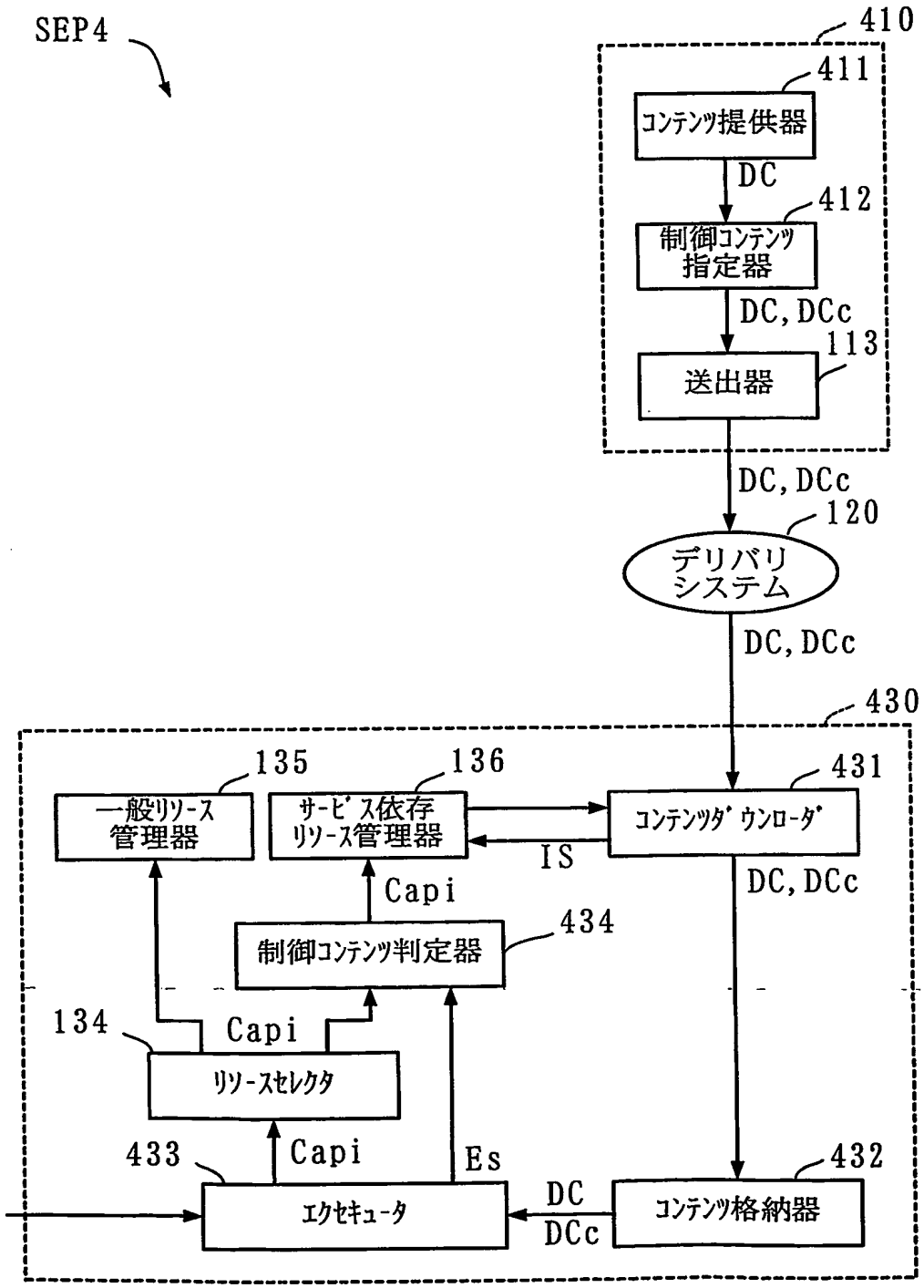

SEP4
410
411
コンテンツ提供器
DC
412
制御コンテンツ指定器
DC, DCc
113
送出器
DC, DCc
120
デリバリシステム
DC, DCc
430
135
一般リソース管理器
136
サービス依存リソース管理器
431
コンテンツダウンローダ
IS
DC, DCc
Capi
434
制御コンテンツ判定器
134
Capi
リソースセレクタ
433
Capi
Es
432
エクセキュータ
DC
DCc
コンテンツ格納器

図１５

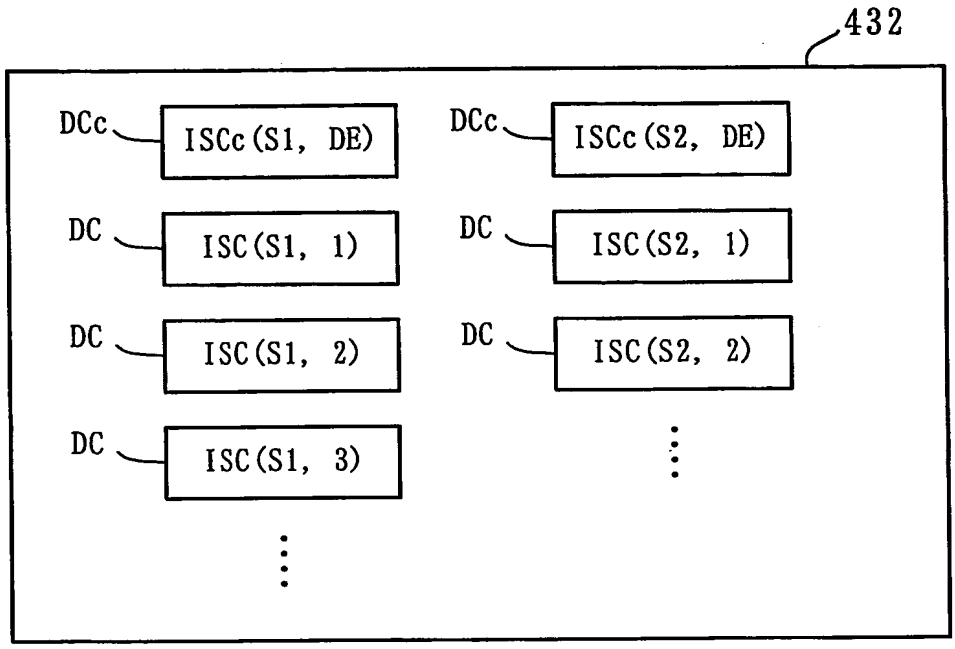

432
DCc
ISCc(S1, DE)
DCc
ISCc(S2, DE)
DC
ISC(S1, 1)
DC
ISC(S2, 1)
DC
ISC(S1, 2)
DC
ISC(S2, 2)
DC
ISC(S1, 3)

図１６

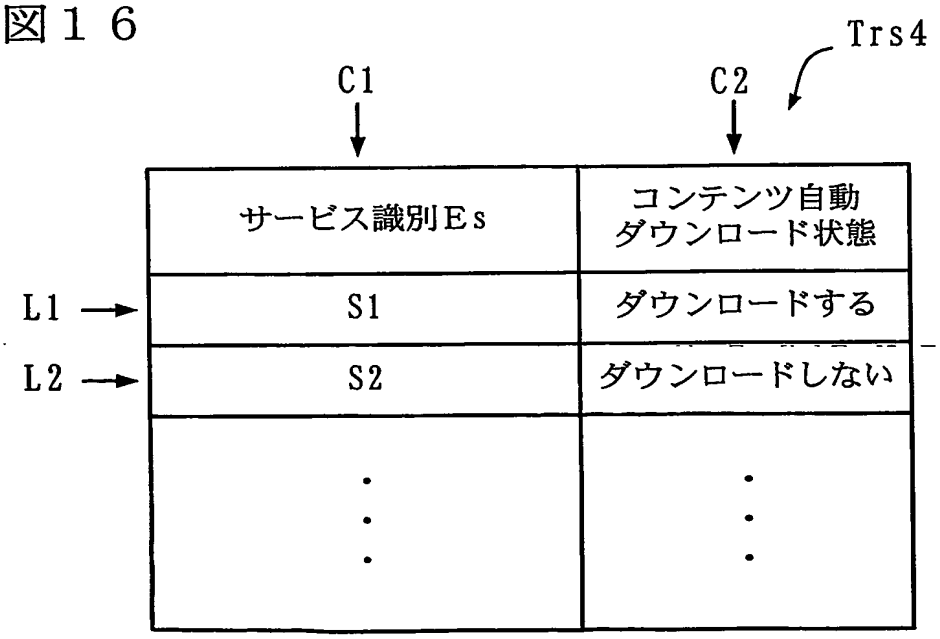

Trs4
C1
C2
サービス識別Es
コンテンツ自動ダウンロード状態
L1
S1
ダウンロードする
L2
S2
ダウンロードしない

図 17

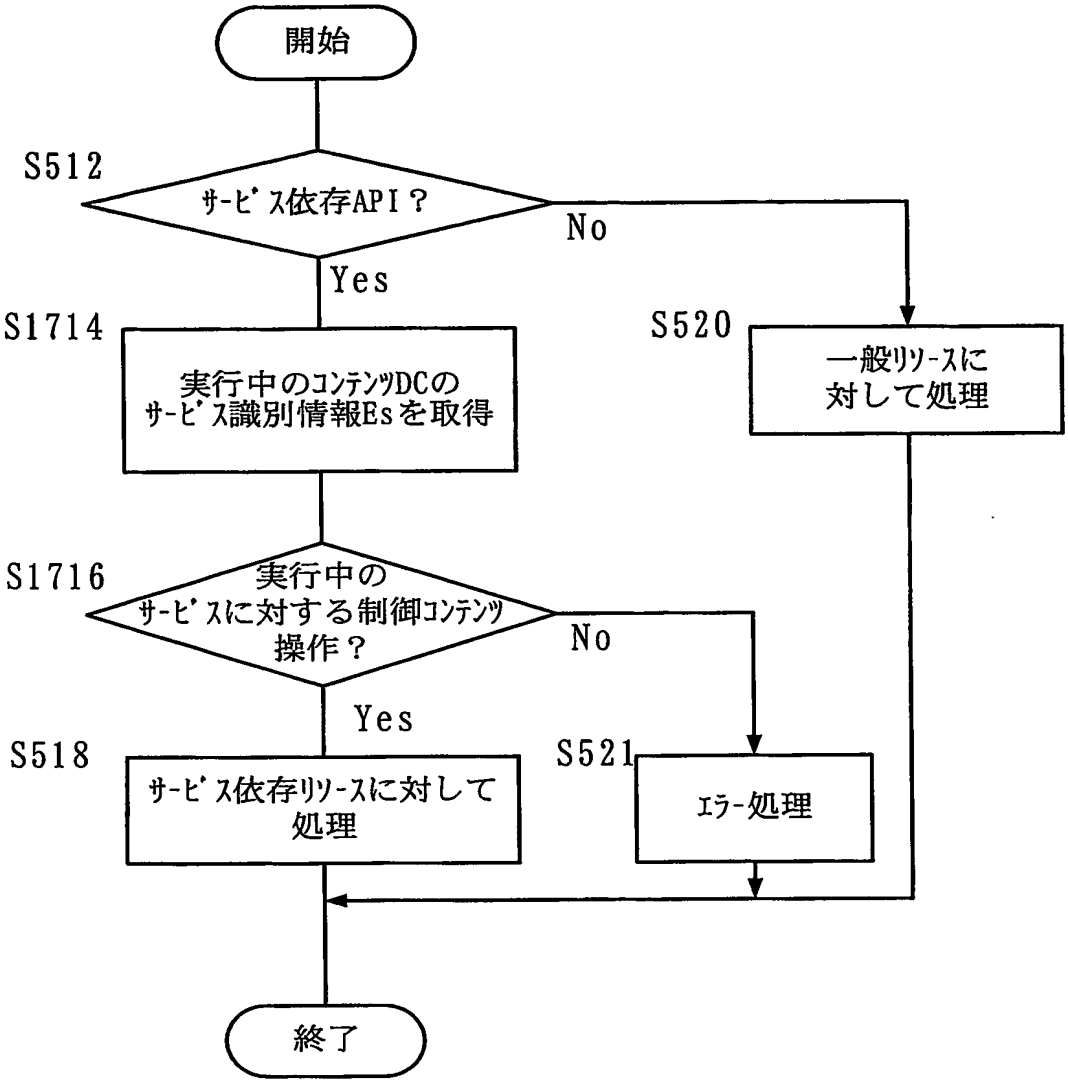

開始
S512
ｻｰﾋﾞｽ依存API？
No
Yes
S1714
実行中のｺﾝﾃﾝﾂDCの
ｻｰﾋﾞｽ識別情報Esを取得
S520
一般ﾘｿｰｽに
対して処理
S1716
実行中の
ｻｰﾋﾞｽに対する制御ｺﾝﾃﾝﾂ
操作？
No
Yes
S518
ｻｰﾋﾞｽ依存ﾘｿｰｽに対して
処理
S521
ｴﾗｰ処理
終了

図 １８

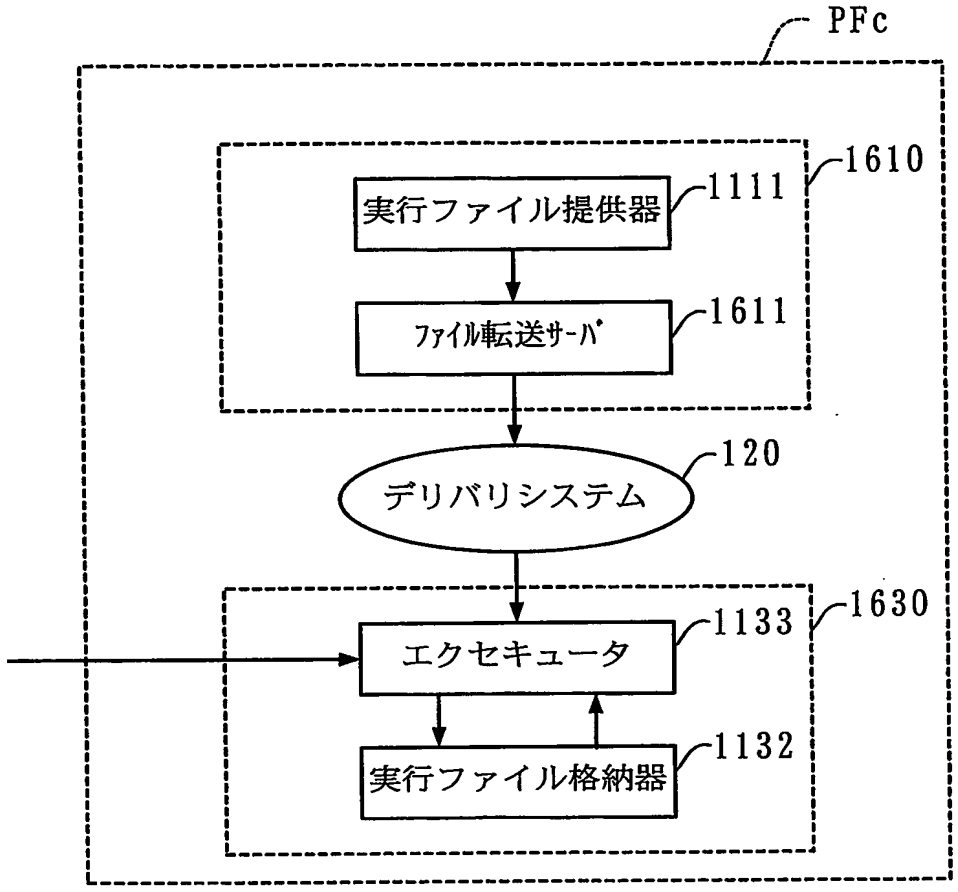

PFc
1610
実行ファイル提供器
1111
ﾌｧｲﾙ転送ｻｰﾊﾞ
1611
120
デリバリシステム
1630
エクセキュータ
1133
実行ファイル格納器
1132

図 １９

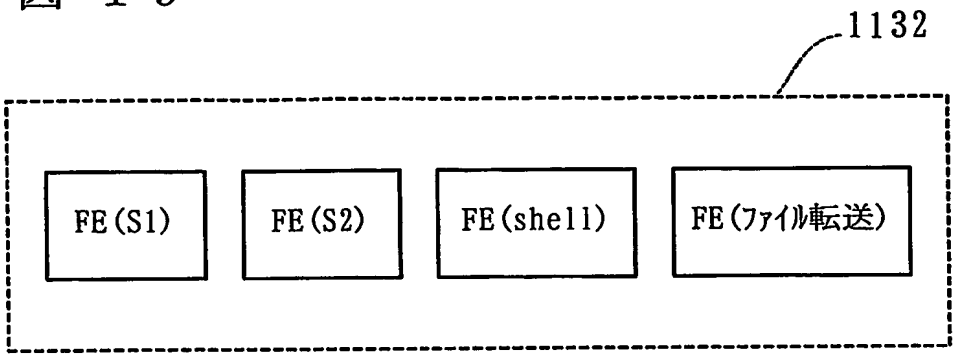
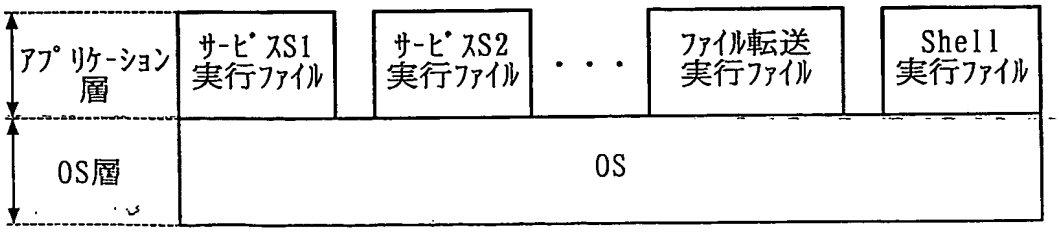

1132
FE(S1)
FE(S2)
FE(shell)
FE(ファイル転送)

図 ２０


アプリケーション層
サービスS1 実行ファイル
サービスS2 実行ファイル
・・・
ファイル転送 実行ファイル
Shell 実行ファイル
OS層
OS

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP03/07390

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[7] G06F9/445

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[7] G06F9/445

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1926-1996	Jitsuyo Shinan Toroku Koho	1996-2003
Kokai Jitsuyo Shinan Koho	1971-2003	Toroku Jitsuyo Shinan Koho	1994-2003

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	JP 2001-5794 A (NEC Corp., Chugoku Nihon Denki Software Kabushiki Kaisha), 12 January, 2001 (12.01.01), Full text; all drawings (Family: none)	1-4 5-12
Y A	JP 2000-172646 A (Toshiba Corp.), 23 June, 2000 (23.06.00), Full text; all drawings (Family: none)	1-4 5-12

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 September, 2003 (16.09.03)	07 October, 2003 (07.10.03)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

国際調査報[redacted]　　国際出願番号 [redacted]T／ＪＰ０３／０７３９０

A.　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Ｉｎｔ　Ｃｌ⁷　Ｇ０６Ｆ９／４４５

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Ｉｎｔ　Ｃｌ⁷　Ｇ０６Ｆ９／４４５

最小限資料以外の資料で調査を行った分野に含まれるもの
日本国実用新案公報　　１９２６年－１９９６年
日本国公開実用新案公報　１９７１年－２００３年
日本国実用新案登録公報　１９９６年－２００３年
日本国登録実用新案公報　１９９４年－２００３年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
Ｙ Ａ	ＪＰ　２００１－５７９４　Ａ（日本電気株式会社，中国日本電気ソフトウェア株式会社）２００１．０１．１２，全文，全図（ファミリーなし）	１－４ ５－１２
Ｙ Ａ	ＪＰ　２０００－１７２６４６　Ａ（株式会社東芝） ２０００．０６．２３，全文，全図（ファミリーなし）	１－４ ５－１２

☐ Ｃ欄の続きにも文献が列挙されている。　　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
１６．０９．０３	07.10.03

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）		
日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	漆原　孝治	５Ｂ	９３６６
	電話番号　０３－３５８１－１１０１　内線　３５４６		

様式ＰＣＴ／ＩＳＡ／２１０（第２ページ）（１９９８年７月）